UNIVERSIDAD POLITÉCNICA DE MADRID

FACULTAD DE INFORMÁTICA

TESIS DOCTORAL

Análisis de expresiones faciales mediante visión por computador

presentada en la
FACULTAD DE INFORMÁTICA
de la
UNIVERSIDAD POLITÉCNICA DE MADRID
para la obtención del
GRADO DE DOCTOR EN INFORMÁTICA

AUTOR: **José Miguel Buenaposada Biencinto**
DIRECTOR: **Luis Baumela Molina**

Madrid, 2005

A Verónica con todo mi amor.

"Me lo contaron y lo olvidé, lo vi y lo entendí, lo hice y lo aprendí"
Confucio

Agradecimientos

En primer lugar, me gustaría agradecer a José Luis Sierra (alias "JL") aquella conversación en la que me ayudó a decidirme a hacer en trabajo fin de carrera con Luis Baumela (seguro que JL ni se acuerda). Sobre todo, porque sin ese cambio de orientación no habría continuado en el mundo de la investigación.

A Luis Baumela tengo que brindarle la tesis (como los toreros) dado que la mayor parte del mérito es suyo. Con Luis he aprendido lo que significa investigar y he descubierto un campo de trabajo fascinante: la Visión por Computadora. Nadie te cuenta al comenzar en esto lo importante que es un buen director de tesis, y tengo que decir que Luis, salvando lo poco que utiliza el látigo (a veces se necesita un poco más de presión), ha sido un magnífico director y un buen amigo.

Por supuesto, los compañeros de trabajo hacen mucho, con ellos se pasa el día a día, se toman cafés, se resuelven dudas y sobre todo se divierte uno trabajando. Enrique Muñoz (alias "Kike") y Javier de Lope (alias "jdlope") han sido esos compañeros y amigos con los que sobre todo comparto la pasión por la informática y las computadoras. Gracias por ser tan "freaks" y/o "geeks". No me puedo olvidar de "las chicas de estadística" (Concha, Arminda y Maribel) con las que he compartido innumerables ratos en las comidas hablando de todo un poco. Espero, Concha, que algún día seamos capaces de escribir ese artículo conjunto. También a mis compañeros de la Universidad Rey Juan Carlos, les agradezco el haberme aguantado en el último año de tesis y, sobre todo, la aportación de Juan Manuel Serrano, Jorge San Martín y Sergio Saugar con su participación en los experimentos de animación facial.

A mis padres les debo el haberme proporcionado todas las oportunidades y la libertad para desarrollarlas. Sin su ayuda ni siquiera podría haberme planteado el iniciarme en el mundo de la investigación. Mi familia me apoyó durante los años de licenciatura y posteriormente durante los primeros años de formación predoctoral aún sin entender demasiado a qué conduce la carrera investigadora en España. Sinceramente, uno todavía no sabe a que conduce, a parte de a una enorme satisfacción personal con el trabajo realizado.

Alguien me dijo una vez que la "profesión" de investigador la elegía yo pero la sufría mi pareja. Y ciertamente nunca podré agradecerle lo suficiente a Verónica el haber apoyado y sufrido mi decisión de dedicarme a esto de la investigación. Ella es la que mejor me comprende y mi mejor amiga. Gracias por todo lo bueno que hemos vivido juntos hasta ahora y sobre todo por el hijo que vamos tener.

Llevar a término la tesis doctoral sólo se puede lograr con la ayuda y el apoyo

de muchísima gente. En estas líneas va mi agradecimiento a todos aquellos que de una forma u otra ayudaron a que me decidiera por el mundo de la ciencia y la tecnología: profesores, amigos, familia e incluso los directores de programas de televisión educativos, que de todo hubo.

Diciembre de 2004.

Índice general

Índice de figuras

Índice de cuadros

Índice de Algoritmos

Resumen

Las expresiones del rostro son una componente esencial de los procesos de comunicación entre los seres humanos. No es sorprendente, por tanto, que la cuantificación de las expresiones faciales constituya un tema de interés en el campo de la visión por computador por su utilidad en la construcción de interfaces avanzadas de comunicación con el ordenador. También es un tema de interés en el campo de la animación, área en la que confluyen las técnicas gráficas y la visión.

La cara es un objeto difícil de analizar mediante técnicas de Visión por Computador pues presenta amplias zonas con poca textura y sus regiones más expresivas (ojos, cejas y boca) sufren deformaciones no rígidas. Si a esto le añadimos la dificultad de modelizar y predecir el movimiento de la cabeza junto a la existencia de oclusiones o de condiciones cambiantes de iluminación, entenderemos por qué el análisis del rostro mediante Visión por Computador es un problema difícil que hoy en día continúa abierto.

En la presente tesis desarrollamos un sistema de análisis facial que nos permitirá encontrar y seguir el rostro humano así como cuantificar sus expresiones faciales. Para el seguimiento utilizaremos una arquitectura basada en el "Enfoque Gradual de la Atención". Este arquitectura está formada por un conjunto de seguidores con distintos niveles de precisión y de carga computacional. El seguidor menos preciso consiste en buscar la cara aleatoriamente en la imagen, prestando atención a regiones con un color parecido al de la piel. Una vez tenemos una región candidata, se emplea la textura de la cara de la persona a seguir para localizar de forma más precisa su cabeza. Un seguidor basado en la apariencia, entrenado con la cara del usuario, nos permitirá tratar el movimiento no rígido del rostro y, al mismo tiempo, estimar el movimiento rígido del conjunto. El sistema resultante controla el coste computacional y la precisión en el seguimiento, empleando un seguidor menos preciso y de menor coste cuando las condiciones del entorno se degradan, y aumentando la precisión cuando mejoran. Finalmente, como aplicación práctica, emplearemos los parámetros de movimiento estimados para animar un modelo gráfico 3D.

Para afrontar el problema de los cambios de iluminación, se ha propuesto un procedimiento de normalización de color basado en un conocido algoritmo de constancia de color para escenas estáticas, el algoritmo Grey World, pero extendido al caso de secuencias de imágenes. Con el nuevo desarrollo se pueden seguir objetos a partir de su color, con un buen grado de robustez a los cambios de iluminación.

Además, se ha desarrollado un seguidor eficiente basado en diferencias en los niveles de gris (SSD) que puede estimar la posición y orientación en 3D de un plano

mediante el empleo de un modelo de movimiento proyectivo. Está basado en la extensión al caso proyectivo de la idea de la factorización del Jacobiano de Hager y Belhumeur. También se ha desarrollado un procedimiento de elección del conjunto de píxeles más informativo de la plantilla de seguimiento para incrementar aún más el rendimiento del seguidor.

Finalmente, se ha desarrollado desarrollado una técnica de factorización del Jacobiano para resolver el problema de seguimiento de objetos deformables. Usando nuestro algoritmo es posible seguir el movimiento no rígido de las zonas de la cara en tiempo real. El algoritmo resultante es interesante no sólo por su eficiencia computacional, sino también porque es más fácil de entrenar que los bien conocidos Modelos Activos de Apariencia (AAM).

Summary

Facial expressions are essential components in human communication processes. Not surprisingly, facial expression quantification is topic of interest in Computer Vision research due to its applicability in advanced interfaces for human-computer interaction. Also it is a very interesting topic in computer animation, an area in which computer graphics and vision converge.

Human face is not an easy object to analyse by using Computer Vision techniques, as it mostly textureless and facial expressions produce non-rigid motion on its most expressive regions (eyes, eyebrows and mouth). If we add the difficulties of modelling and predicting head motion and even the inconvenients that arise with occlusions or illumination changes, we can understand why facial analysis using Computer Vision is a difficult and still open problem.

In this thesis we develop a facial analysis system that will track the human face and quantify its facial expressions. For tracking we will use an "Incremental Focus of Attention" architecture. This architecture is made of a set of trackers with different levels of precision and computation load. The lowest precision tracker searchs randomly for the face in the image, looking for regions with skin-like colour. Once we have a candidate image region, the face texture of the user is used to accurately localise the head. An appearance based tracker, trained with the user's face, deals with the non-rigid motion of the face and, at the same time, estimates the rigid motion of the full head. The system is able to control the computational load and the tracking precision by using the lower precision and lower computational load tracker when environment conditions get worse and rising the precision when conditions gets better. Finally, as an application, we will use the motion parameters estimated to animate a 3D graphical model of the human head.

To deal with illumination changes, we have extended to images sequences a well known colour constancy algorithm for static scenes, the Grey World algorithm. With the new development we can track a coloured object with light colour changes.

We have also developed an efficient SSD based tracker that can estimate the 3D position and orientation of a plane. It is based on an extension to the projective motion model of the Hager and Belhumeur's Jacobian Matrix factorisation idea. Also, we have developed a procedure to choose the most informative set of pixels over the template image to increase even further the performance of the tracker by mean of reducing the number of pixels needed.

Finally, we have developed a Jacobian factorisation technique to solve the deformable objects tracking problem. By using our new algorithm it is possible to track

the non rigid motion of face regions in real-time. The resulting algorithm is interesting not only because of its real-time performance, but also because it is easier to train than the well known Active Appearance Models.

Parte I

Introducción

Capítulo 1

Introducción y objetivos

Las expresiones del rostro, el lenguaje corporal y la voz constituyen las vías de comunicación naturales entre los seres humanos. Sin embargo, los métodos de interacción persona-ordenador más extendidos hoy en día, el ratón y el teclado, distan mucho de parecerse a los empleados en la comunicación humana. En búsqueda de una interacción persona-ordenador más natural las interfaces de usuario *multimodales* [100] pretenden emplear diferentes "modos" de comunicación: reconocimiento de voz, análisis de expresiones faciales, escritura a mano alzada, lectura de labios, etc. En este contexto el análisis de expresiones faciales puede convertirse en un elemento clave, dado que es fundamental en multitud de aplicaciones que necesiten de la interacción con el usuario: ayuda a personas discapacitadas (p.ej. guiado de sillas de ruedas mediante un lenguaje de comandos basado en expresiones faciales), vídeo-conferencia empleando un ancho de banda reducido (combinando codificación de expresiones faciales y la animación por ordenador), animación facial en infografía, estudios del comportamiento del conductor (mediante la estimación de la dirección de la mirada), estudios de la reacción del consumidor ante anuncios de televisión, controles de seguridad biométricos en el acceso a edificios, lectura de labios como apoyo al reconocimiento de voz en ambientes inherentemente ruidosos (un bar, un tren, un avión, etc), etc.

Dada la potencia de cálculo de las computadoras personales, el bajo coste de las cámaras digitales y madurez de muchas de las técnicas de la Visión por Computador, esta última es posiblemente la aproximación más prometedora para para construir un sistema automático de análisis facial. De hecho, desde hace algunos años la industria cinematográfica y la televisión ya vienen empleando, con gran éxito (véase películas como "Final Fantasy" o la saga de "El Señor de los anillos"), sistemas de captura de movimiento facial basados en técnicas de Visión por Computador que emplean marcadores reflectantes (p.ej. el sistema Vicon [53]) para localizar y seguir las partes más expresivas de la cara. El problema con la mayoría de sistemas de análisis facial comerciales es que el actor debe pegar en su piel los marcadores, vestir un traje con ellos, o incluso someterse a sesiones de maquillaje e iluminación especiales. Sin embargo, un buen sistema de análisis facial debería ser no invasivo, de forma que el usuario no tuviese que emplear marcadores artificiales en su cara ni ningún otro

dispositivo especial.

La mayoría de las técnicas de Visión por Computador son no invasivas, en el sentido de que procesan una secuencia de imágenes de la cara del usuario sin marcadores ni iluminación especial. Ahora bien, el rostro es un objeto difícil de tratar mediante técnicas de Visión por Computador pues presenta amplias zonas con poca textura y sus regiones más expresivas (ojos, cejas y boca) sufren deformaciones no rígidas. Si además, pretendemos que sea robusto frente a las oclusiones de la cara, cambios en la iluminación o cualquier tipo de movimiento de la cabeza, entenderemos por qué el análisis facial mediante Visión por Computador es un problema difícil que hoy en día continua abierto.

El mejor reflejo de lo interesante del problema, es la inclusión de la codificación de expresiones faciales en el nuevo estándar internacional de compresión de vídeo MPEG-4 [39]. Y por supuesto, otra indicación de lo relevante del problema es la existencia de algunas empresas que comercializan sistemas de análisis facial: Seeing Machines (http://www.seeingmachines.com), Face2Face (http://www.f2f-inc.com/), Eyematic (http://www.eyematic.com), See Storm (http://www.seestorm.com) o A4Vision (http://www.a4vision.com).

Como ya hemos visto en las posibles aplicaciones, el análisis automático y la síntesis de expresiones faciales constituyen un problema relevante no sólo en el campo de la Visión sino también en el de los Gráficos por Computador (de hecho en la parte III de esta tesis presentaremos un ejemplo de la confluencia de ambas disciplinas). En la presente tesis pretendemos estudiar, y mejorar, algunas técnicas de Visión por Computador con el objetivo de construir un sistema no invasivo de análisis facial. Ahora bien ¿en qué consiste el análisis facial?

1.1. Procesamiento visual del rostro

Aunque muy atractiva desde el punto de vista de las aplicaciones, la cara no es un objeto sencillo de analizar empleando técnicas de Visión por Computador. Principalmente debido a la naturaleza no rígida del movimiento de algunas de sus partes más expresivas (boca, mandíbula, ojos y cejas) y a la existencia de grandes zonas sin textura.

En el planteamiento que haremos en esta tesis, tendremos que resolver tres subproblemas:

1. **Búsqueda y seguimiento de la cara**. El seguimiento visual consiste en la estimación, de todos o de parte, de los 6 grados de libertad, posición y orientación, de un objeto mediante el procesamiento de una secuencia de imágenes. Para abordar este problema, en la literatura se ha utilizado gran variedad de elementos de seguimiento: el color de la piel [104]; el contorno hombros y cabeza [56]; el contorno de la cabeza; el movimiento del rostro completo [41]; zonas muy particulares de la cara como pueden ser las comisuras de los labios, los agujeros de la nariz o las pupilas [35]; o la estructura tridimensional de la cabeza en su conjunto [87].

La estimación de la orientación y posición de la cabeza supone la localización del rostro en el plano imagen (ver figura 1.1). Según el número de grados de libertad que estiman los algoritmos de seguimiento visual, estos pueden clasificarse como [93]:

- *Seguimiento 2D*. Realizan el seguimiento sólo de la posición del rostro sobre el plano imagen.
- *Seguimiento 21/2D*. En este grupo están incluidos aquellos algoritmos que realizan un seguimiento 2D con alguna información de orientación del rostro.
- *Seguimiento 3D*. Estiman los seis grados de libertad de la cabeza: orientación y posición en 3D.

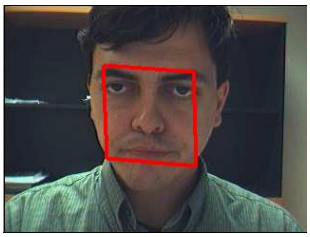


Figura 1.1: Localización y seguimiento del rostro.

2. **Localización de elementos de interés**. Las partes más informativas del rostro son la boca, los ojos, las cejas y la nariz. Una vez localizada la cara en la imagen, el siguiente problema a resolver será la localización de sus partes más expresivas (ver fig. 1.2).

   Este proceso de localización puede llevarse a cabo de dos formas:

   - *Como parte del proceso de seguimiento*. En este caso se emplearán los ojos, los agujeros de la nariz o las comisuras de los labios para seguir el rostro. Su localización se conocerá desde el momento en el que se conozca la posición de la cabeza. En el seguimiento basado en la apariencia , por ejemplo, se emplean los ojos y la boca para localizar la cara en la imagen (ver capítulo 6).
   - *Como un procedimiento separado*. Cuando se emplea, por ejemplo, el color de la piel para seguir la cara, siempre tendremos que localizar los ojos, la boca o la nariz por otros medios.

3. **Clasificación y codificación de expresiones faciales**. Por último, cuando ya sabemos dónde se encuentran las partes interesantes del rostro en la imagen, es el momento en el que comienza el verdadero análisis facial (ver fig.1.3). De una forma orientada a la aplicación deberemos:

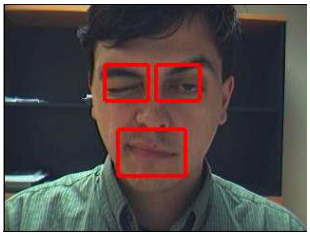


Figura 1.2: Localización de los elementos de interés de la cara.

a) Elegir el sistema de codificación de expresiones faciales, p.ej. FACS, MPEG-4 FAPS, un código *ad hoc*, etc.

b) Encontrar la forma de relacionar los parámetros provenientes del seguimiento del rostro con el sistema de codificación de expresiones faciales elegido.

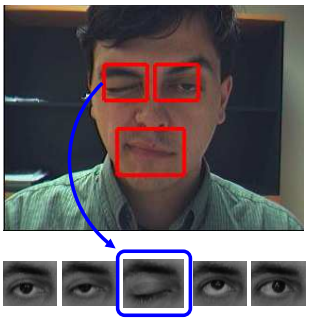


Figura 1.3: Clasificación y codificación de expresiones faciales.

1.2. Objetivos de la tesis

El propósito de la presente tesis es el estudio del problema de "análisis de expresiones faciales mediante Visión por Computador". El sistema que desarrollaremos tendrá como entrada una secuencia de imágenes proveniente de una cámara (o almacenada en disco) y su salida será la posición del rostro en cada imagen junto con la cuantificación de la deformación de ojos, cejas y boca producidas por la expresión del rostro. Además hemos impuesto dos requisitos fundamentales para que el sistema sea útil:

- Robustez ante los cambios en el entorno de funcionamiento (cambios de iluminación, oclusiones, etc.).
- Funcionamiento en tiempo real (25/30 imágenes por segundo).

El *cómo* lograr estas metas es lo que desarrollaremos en los capítulos que siguen.

1.3. Requisitos de la solución

De lo expuesto en la sección 1.1 podemos deducir que existen multitud de formas de abordar el análisis de expresiones faciales. Se puede emplear una cámara o un conjunto de ellas, un sistema de visión calibrado o sin calibrar, se puede emplear el color de la piel o marcas en el proceso de seguimiento, etc. Por eso es necesario definir cuál es el problema que queremos resolver y qué restricciones impondremos a la solución. Es importante establecer a priori cuáles son los requisitos de una solución válida al problema de seguimiento que tenemos entre manos. En nuestro caso hemos optado por (ver fig. 1.4):

- **Una única cámara**. La mayor parte de los usuarios de una computadora o un teléfono móvil únicamente disponen de una cámara, lo que supone una buena razón para adoptar esta restricción.

- **Imágenes frontales/o casi frontales**. Esta situación se dará cuando el usuario sea el de una computadora de sobremesa con la cámara sobre el monitor o un teléfono móvil al que el usuario sostiene entre sus manos mientras observa su pantalla.

- **Imágenes de los hombros y cabeza**. Este requisito únicamente supone que la cara ocupará una determinada porción de la imagen y que la mayor parte del tiempo las manos estarán ocultas a la cámara.

- **El sistema se adaptará al usuario mediante aprendizaje**. Dado que es mucho más fácil el seguimiento de un objeto que se conoce previamente y del que tenemos un modelo, emplearemos un modelo de cada usuario. Para construirlo se empleará una (o varias) secuencia(s) de imágenes que servirán como datos de entrada al proceso de entrenamiento del sistema.

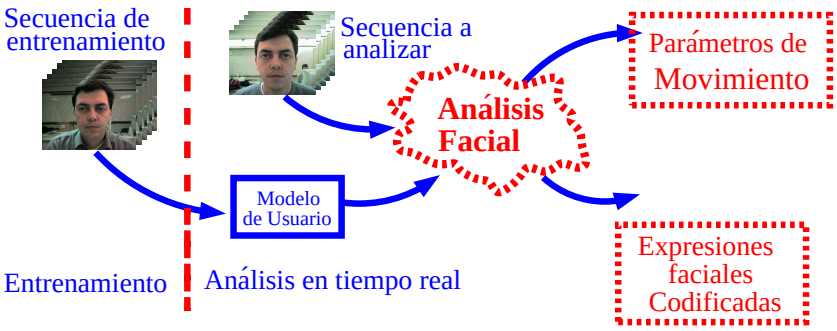


Figura 1.4: Planteamiento del problema de análisis facial.

1.4. Arquitectura del sistema

Todo algoritmo que podamos utilizar en el seguimiento visual de un objeto, falla en alguna de las situaciones que nos encontramos en la vida cotidiana: cambios del color de la iluminación o de su intensidad, oclusiones, un fondo complicado (o en el caso extremo constituido por objetos con las mismas características que el objeto seguido), movimiento muy rápido, etc. Por muy bueno que sea un algoritmo de seguimiento, siempre habrá situaciones en la que su trabajo se desarrollará con dificultades y, en el caso extremo, perderá al objeto seguido. Una posible solución a este problema es la utilización coordinada de diferentes algoritmos de seguimiento [56, 93]. Para lograr mayor robustez, será mejor emplear más de un seguidor con condiciones de fallo no coincidentes. Por ejemplo, un seguidor basado en el movimiento perderá al objeto cuando se quede inmóvil, sin embargo, si lo utilizamos junto a un seguidor del contorno de la cara podremos continuar el seguimiento con este último.

Nuestra solución se basa en una arquitectura que emplea un conjunto de algoritmos de seguimiento y un autómata que coordina su ejecución. Esta filosofía de trabajo permitirá al conjunto responder de una manera flexible a las variaciones de las condiciones de funcionamiento (iluminación, oclusiones, etc.) obteniendo en cada momento toda la información disponible sobre la posición y orientación del objeto seguido. Cuando las condiciones de trabajo sean óptimas, seguirá con precisión el movimiento del rostro y, a medida que estas condiciones se deterioren, el seguimiento se hará menos preciso. El sistema funcionará dinámicamente de forma que el autómata seleccionará en cada momento el algoritmo de seguimiento más preciso posible.

La arquitectura está organizada en cuatro niveles, o seguidores, con un autómata finito coordinando la ejecución (ver Fig. 1.5). Cada nivel representa un seguidor que emplea diferentes elementos de seguimiento.

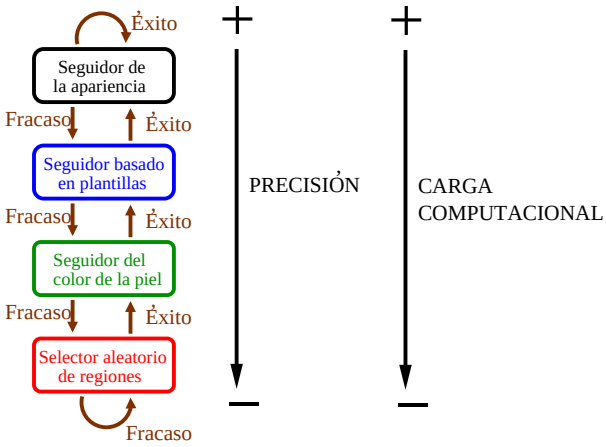


Figura 1.5: Arquitectura de seguimiento del sistema desarrollado.

El primer nivel es un selector de regiones aleatorio. Cuando el sistema se encuentra en el momento inicial o ha perdido al objetivo, entonces busca aleatoriamente la cara en el campo de visión de la cámara.

El segundo nivel es un seguidor basado en el color. Emplea una versión modificada de un algoritmo de constancia del color, *Grey World* o de la *hipótesis del mundo gris*, para modelar el color de la piel.

El siguiente nivel es un seguidor basado en flujo óptico parametrizado, que estima la posición y orientación en 3D de la cabeza tomando como modelo una imagen del usuario.

Finalmente, las expresiones faciales del usuario se cuantifican a través de un seguidor basado en la apariencia. Con él se estiman la posición y orientación del rostro en su conjunto (movimiento rígido) y se cuantifica la deformación de ojos, boca y cejas.

En los capítulos que siguen presentaremos los algoritmos que forman nuestro sistema de análisis de expresiones faciales.

1.5. Estructura de la memoria

La memoria se ha organizado en cuatro partes que, comenzando con la presente introducción (parte I), presentan los diferentes niveles o técnicas empleadas en nuestro sistema de análisis facial, desde el seguimiento a bajo nivel basado en el color de la piel (parte II), hasta la reanimación de expresiones faciales (parte IV), pasando por la localización precisa del rostro en la imagen (parte III).

En la parte I, se ha introducido el problema del análisis facial, se han establecido los objetivos de la tesis doctoral, así como la arquitectura del sistema desarrollado.

En la parte II, se estudia el seguimiento basado en el color de la piel: se realiza una revisión bibliográfica (capítulo 2) y se propone una extensión a secuencias de vídeo de un algoritmo de constancia del color para imágenes estáticas (capítulo 3).

La parte III, comienza con una revisión de los métodos directos (aquellos que emplean directamente los niveles de gris para estimar el movimiento), en el seguimiento de regiones planas (capítulo 4). Continúa con nuestra extensión de la factorización del Jacobiano para el caso proyectivo (capítulo 5). Y termina con un nuevo algoritmo para el seguimiento de objetos que cambian de apariencia (capítulo 6).

En la parte IV, se trata el problema de la reanimación de un modelo gráfico 3D a partir de la información proporcionada por el seguidor basado en la apariencia (capítulo 7).

Y por último, en la parte V se exponen las conclusiones así como las líneas de trabajo que han quedado abiertas durante el desarrollo de la presente tesis doctoral (capítulo 8).

Parte II

El color de la piel como elemento de seguimiento

Capítulo 2

Seguimiento basado en el color

La segmentación de imágenes basada en el color [83] ha recibido una gran atención como medio para realizar seguimiento de objetos [93]. La razón principal es que los seguidores basados en esta característica son rápidos y fiables, aunque no muy precisos. Se emplean como estimación inicial de la posición del rostro en el plano imagen, o como procedimiento de recuperación cuando los seguidores más precisos no pueden continuar.

A pesar de estas ventajas del color como información visual, no podremos evitar confundirnos con los objetos de color similar al que buscamos [72, 93]. Por otro lado, con la información que nos proporciona no podremos estimar más que la posición del objeto seguido y, a lo sumo, su orientación en el plano imagen. De lo que se deduce que empleando únicamente la información cromática como elemento de seguimiento no tendremos la solución completa al problema de seguimiento visual.

Salvando las dificultades inherentes a la información cromática, es sencillo diseñar un algoritmo de seguimiento basado en el color que funcione correctamente en un rango limitado de iluminaciones [12, 59]. El verdadero problema consiste en el funcionamiento con diferentes iluminaciones, ya que bajo dos fuentes de luz distintas los valores RGB[1] registrados por una cámara y pertenecientes a una misma superficie pueden ser muy diferentes [38].

En lo que sigue se introducirán los conceptos más importantes para el seguimiento en color, los problemas a resolver para obtener un seguimiento suficientemente robusto basado en esta característica, así como diversas soluciones adoptadas en la literatura.

2.1. Formación de imagen en color

Para poder establecer qué valores RGB de un píxel pertenecen o no al color de la piel, o a cualquier otro, es necesario conocer cómo se obtiene y qué factores afectan al color percibido en los sensores de la cámara. En esta sección vamos a introducir

[1] Emplearemos el acrónimo inglés de Rojo, Verde y Azul, RGB, por ser la notación más extendida en la literatura

el modelo de formación de imágenes en color conocido como *integración espectral* y que es aplicable a los sensores RGB de cada píxel. Si tomamos como iluminación un foco de luz puntual, y suponiendo que las superficies de la escena son *Lambertianas*[2] tendremos [38]:

$$\bar{p}_{x,E} = \bar{e}_x^\top \bar{n}_x \int_\omega E(\lambda) S_x(\lambda) F(\lambda) d\lambda, \tag{2.1}$$

lo que significa que el color en los sensores R, G o B, en un píxel x, $\bar{p}_{x,E}$, depende de la función de respuesta del sensor, $F(\lambda)$, en todas las longitudes de onda de la luz visible, ω, de la reflectancia de las superficies que aparecen en la escena, $S_x(\lambda)$, y de la composición espectral de la fuente de luz, $E(\lambda)$, o lo que es lo mismo, de la intensidad y del color de la misma [34, 38]. La potencia de la luz reflejada es gobernada por el producto escalar $\bar{e}_x^\top \bar{n}_x$. Donde $\bar{n}_x$ es el vector unitario correspondiente a la normal a la superficie y $\bar{e}_x$ es la dirección de la fuente de luz. El módulo de $\bar{e}_x$ modela la potencia de la luz incidente en el píxel x. Esto implica que $E(\lambda)$ sea constante a lo largo de toda la escena.

Un material parece blanco si su reflectancia es la unidad en todo el espectro visible [70]. De la ecuación 2.1 se puede deducir que los ratios entre R, G, y B cambiarán cuando se cambie a una fuente de luz con otro color (cambios en $E(\lambda)$). Ésta es la razón por las que la cámaras de vídeo tienen que hacer balance de blancos siempre que cambia el color de la fuente de luz. El balance de blancos se hace tomado una imagen de una superficie con reflectancia uno (por ejemplo, Sulfato de Bario) y ajustando los tres canales de color de forma que se cumpla R = G = B. En adelante, siempre que hablemos de un cambio en el color de la iluminación será equivalente a un cambio con respecto al color de la luz con la que se hizo el balance de blancos de la cámara.

Sustituyendo la integral de la ecuación (2.1) por el vector $\bar{q}_{x,E}$ (constante si lo es la composición espectral del foco de luz, esto es, el color de la iluminación), tendremos:

$$\bar{p}_{x,E} = (\bar{e}_x^\top \bar{n}_x) \bar{q}_{x,E}. \tag{2.2}$$

Si mantenemos constante el color de la luz incidente (la composición espectral de la misma) y no movemos los objetos de la escena (los $S_x(\lambda)$ son constantes), entonces $\bar{p}_{x,E}$ será constante en la ecuación (2.2). Diremos que se ha producido un cambio en la geometría del sistema de iluminación cuando $\bar{q}_{x,E}$ permanezca constante y varíe $\bar{e}_x$ ya sea en módulo (un cambio en la intensidad de la luz) o en dirección (un movimiento de la fuente de luz). Los valores R, G y B de un píxel determinado quedan multiplicados por el mismo factor $s_x = \bar{e}_x^\top \bar{n}_x$. Este valor en general será diferente para cada píxel. Por ejemplo, si tomamos una cara iluminada desde el lado izquierdo se percibirá mayor intensidad en los píxeles de la cara que se encuentran en ese lado y progresivamente un factor de escala menor a medida que nos vamos desplazando hacia el lado derecho de la misma. De la misma forma un cambio en la intensidad

[2]En ellos la intensidad de la luz reflejada sólo depende del ángulo de incidencia de la luz y de la normal a la superficie. Constituyen unas características ideales que se cumplen aproximadamente en las superficies mate.

de la luz en la escena modificará el valor de s_x debido al cambio en el módulo del vector $\bar{e}_x$.

Estudiemos ahora el efecto de un cambio en el color de la iluminación. En este caso mantendremos constante $\bar{e}_x$ y, para simplificar los desarrollos posteriores, supondremos que los sensores de la cámara son funciones delta: $F(\lambda) = \delta(\lambda - \lambda_i)$ con $i = (1, 2, 3)$. La respuesta de la cámara con una iluminación dada por $E(\lambda)$ será:

$$\bar{p}_{i,x,E} = \bar{e}_x \cdot \bar{n}_x \int_\omega E(\lambda) S_x(\lambda) \delta(\lambda - \lambda_i) d\lambda = \bar{e}_x \cdot \bar{n}_x E(\lambda_i) S_x(\lambda_i) \tag{2.3}$$

y con la iluminación dada por $E_1(\lambda)$ será:

$$\bar{p}_{i,x,E_1} = \bar{e}_x \cdot \bar{n}_x \int_\omega E_1(\lambda) S_x(\lambda) \delta(\lambda - \lambda_i) d\lambda = \bar{e}_x \cdot \bar{n}_x E_1(\lambda_i) S_x(\lambda_i). \tag{2.4}$$

Combinando (2.3) y (2.4) obtenemos:

$$p_{i,x,E_1} = \frac{E_1(\lambda_i)}{E(\lambda_i)} p_{i,x,E}, \tag{2.5}$$

lo que significa que cuando cambia el color de la iluminación, los valores RGB registrados se multiplican por un factor (uno diferente para cada canal). Aunque la asunción de funciones delta como respuesta de los sensores de la cámara no es aplicable en general, algunos estudios han mostrado que la mayoría de los sensores se comportan, o se puede hacer que se comporten, como funciones delta [38]. El resultado es que si $\underline{\mathrm{R}}$, $\underline{\mathrm{G}}$ y $\underline{\mathrm{B}}$ denotan todos los valores registrados en una imagen (para cada uno de los canales rojo, verde y azul) entonces ante un cambio del color de la iluminación (si se mantiene constante la geometría de la misma) la imagen capturada se convierte en $\alpha\underline{\mathrm{R}}$, $\beta\underline{\mathrm{G}}$ y $\gamma\underline{\mathrm{B}}$ (donde α,β y γ son escalares). Y lo que es más importante: estos coeficientes son los mismos para todos los píxeles de la imagen.

De la influencia en los valores RGB de la geometría y el color de la iluminación se deriva el "modelo dicromático de difracción" [34, 38] para superficies *Lambertianas*. Los valores RGB pertenecientes a un objeto mate bajo cualquier iluminación se distribuyen según una línea que aproximadamente pasa por el origen y cuyo vector director viene dado por (α, β y γ). Los brillos, sin embargo, provocan que lo que debería constituir una línea se convierta en una "T" o una "L" (que se encontrará en el plano dicromático), según el modelo dicromático de refracción. En el caso de la piel humana, debido a que únicamente refleja el 5 % de la luz que incidente [71] (minimizándose el número de píxeles saturados por la aparición de un brillo) la agrupación en el espacio RGB toma una forma muy cercana a una esfera y la forma de "T" o "L" prevista por el modelo, aparece muy desfigurada.

Aún conociendo cómo afecta la iluminación a los valores RGB pertenecientes al color de la piel, no podemos eliminar la influencia de los coeficientes α, β y γ porque su valor es desconocido. La determinación de los coeficientes que transforman los valores RGB de una superficie tomados bajo una iluminación, en los valores correspondientes a la misma superficie vistos bajo una iluminación de referencia constituye

el problema *constancia del color*. Se trata de un problema difícil de resolver con escenas en las que no aparece el número suficiente de superficies de color diferente [32].

2.2. Constancia del color en el seguimiento del rostro

Desde un punto de vista biológico, la constancia del color también puede definirse como la habilidad perceptual para asignar el mismo color a los objetos bajo diferentes condiciones de iluminación. La meta de cualquier algoritmo de constancia de color es transformar los valores $[RGB]$ de la imagen en descriptores de color constantes. Lo que se traduce en 1) encontrar el color de la iluminación ya sea en RGB o en otro espacio cromático y 2) establecer la relación existente entre el blanco bajo esta iluminación y el mismo bajo una iluminación de referencia para poder transformar los descriptores de color de la imagen actual a la iluminación de referencia [31].

Como vimos en la sección 2.1, si la intensidad de la luz se multiplica por un factor s, entonces el color percibido en cada píxel i se convierte en $[s_iR_i, s_iG_i, s_iB_i]$. El espacio cromático rg (o normalización RGB), proporciona una solución a la constancia del color que es independiente de la intensidad y geometría (la dirección y posición de la fuente de luz) de la iluminación:

$$[s_iR_i, s_iG_i, s_iB_i] \mapsto [\underbrace{\frac{s_iR_i}{s_i(R_i+G_i+B_i)}}_{r_i}, \underbrace{\frac{s_iG_i}{s_i(R_i+G_i+B_i)}}_{g_i}].$$

y es similar a otros muchos empleados en la literatura, por ejemplo, $(R,G,B) \mapsto (\frac{R}{B}, \frac{G}{B})$, $(H,S,V) \mapsto (H,S)$, $(L,U,V) \mapsto (L,U)$, CIE-(x,y), etc. Sin embargo, un cambio en el color de la iluminación se puede modelar como un factor de escala α, β y γ en cada uno de los canales de color de la imagen, R, G, y B. En este caso la normalización anterior falla.

La *hipótesis del mundo gris* (en inglés *Grey world*) [38], (GW) , proporciona una solución a la constancia del color independiente del color de la luz mediante la división de cada canal de color por su media:

$$[\alpha R_i, \beta G_i, \gamma B_i] \mapsto [\frac{\alpha R_i}{\frac{\alpha}{n}\sum_i R_i}, \frac{\beta G_i}{\frac{\beta}{n}\sum_i G_i}, \frac{\gamma B_i}{\frac{\gamma}{n}\sum_i B_i}].$$

aunque desafortunadamente sólo funciona con imágenes estáticas.

El algoritmo de constancia del color más usado en el seguimiento del rostro es el espacio cromático rg [69]. La distribución de probabilidad del color de la piel en el espacio de cromaticidades rg es una normal, y por tanto se puede emplear un clasificador Bayesiano (como veremos en la sección 2.3) para seguir una cara que se mueve [60]. Sin embargo, si cambiamos el color de la iluminación la distribución del color en el espacio rg cambia. Por ello, en la literatura se han propuesto variaciones a este

algoritmo para poder tratar pequeños cambios en el color de la luz. Estas consisten en el seguimiento del movimiento de la normal del color de la piel en el espacio de color mediante el uso de modelos estocásticos de predicción [104] (aunque en este caso se trabaja en el espacio de color (h, s)) o mediante técnicas de agrupamiento (*clustering*) [103, 7]. Estos algoritmos están basados en la hipótesis de suavidad de los cambios de la luz, y por tanto, fallan cuando se incumple esta premisa.

Los algoritmos de constancia del color basados en modelos físicos, utilizan el conocimiento sobre cómo se manifiestan en la imagen procesos como la reflexión de la luz, y la sensibilidad de la cámara. Muchos de estos algoritmos se basan en el modelo dicromático de reflexión [34].

Un método conocido para la constancia del color consiste en encontrar el color de la iluminación como la intersección de al menos dos planos dicromáticos en el espacio RGB [66, 28]. Este método asume la presencia en la imagen de muchas superficies, algo que no es siempre cierto. Finlayson [31] plantea un procedimiento que trabaja en el espacio rg modelando el rango de posibles iluminaciones como el lugar Planckiano de la cromaticidades de un cuerpo negro incandescente a diferentes temperaturas. La cromaticidad de la iluminación se encontrará como la intersección de la recta dicromática (resultado de la intersección del plano dicromático con el plano de vector característico (1,1,1) - normalización RGB -) de una única superficie de color (ver figura 2.1) con el lugar geométrico de las iluminaciones (con forma próxima a una parábola).

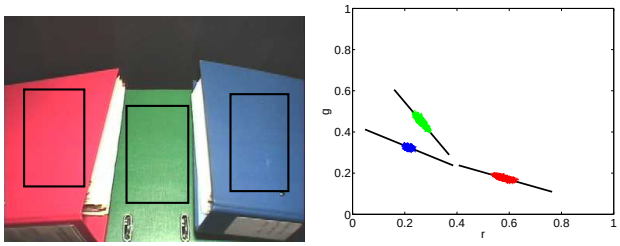


Figura 2.1: Rectas dicromáticas para diferentes superficies en el espacio cromático rg. Las regiones encuadradas en negro en la imagen de la izquierda proporcionan los píxeles que aparecen en la gráfica de la derecha (los colores con los que se representan las nubes de puntos en rg, se corresponden con el color del objeto).

En el trabajo de Störring [71] se estudia el algoritmo de constancia del color de Finlayson [31], cuando la única superficie en la escena es la piel humana. El algoritmo consistirá en este caso, en encontrar el punto de corte entre la recta dicromática de la piel y el lugar geométrico de las iluminaciones. Los resultados para la piel en la estimación de la cromaticidad de la luz incidente no son demasiado buenos por varios motivos: la piel es muy acromática y por tanto sus cromaticidades se encuentran muy cercanas al lugar geométrico de las iluminaciones. Por otro lado, su línea dicromática está lejos de ser ortogonal al lugar geométrico de las iluminaciones (por lo que un pequeño error en la estimación de la línea puede dar un punto de

corte muy alejado del real). La solución de Störring pasa por un modelo que permite encontrar la concentración de melanina en la piel, dada una imagen tomada con una luz de color conocido. El valor de la concentración de melanina permite encontrar en el espacio rg las rectas dicromáticas de la piel con la misma concentración de melanina. La intersección de la recta dicromática con el lugar Planckiano permite encontrar la cromaticidad de la iluminación. El requisito del algoritmo es la disponibilidad de una muestra de píxeles pertenecientes a la piel tomada con un color de la iluminación conocido y, para el modelo físico que se utiliza la disponibilidad de curvas de sensibilidad espectral de la cámara. También son necesarias curvas espectrales de reflectancia de todos los tipos de piel (de la blanca a la negra) en función de la concentración de melanina. La principal dificultad en la aplicación de estos métodos es la medición de todas las curvas espectrales necesarias para el modelo físico de los valores RGB de la piel.

Lo más relevante de los trabajos de Störring [70, 71, 72] es que establece un modelo del color de la piel, validado empíricamente, que es capaz de calcular la zona en la que se encontrarán las cromaticidades de la piel si se conoce el color de la iluminación. Este modelo también es válido cuando tenemos una mezcla de fuentes de luz de color conocido en la escena. Störring estudió también cómo cambia la distribución de píxeles de la piel en el espacio rg cuando lo hace el color de la iluminación. Al cambiar el color de la iluminación, la distribución de píxeles de la piel en el espacio rg se desplaza en una zona muy cercana al lugar geométrico de las iluminaciones siguiendo su curvatura (recordemos que este lugar geométrico tiene forma de parábola).

2.3. Seguimiento del color de la piel

El color de la piel de un individuo viene determinado por una combinación de sangre (rojo) y melanina (amarillo, marrón). Esta combinación produce sólo un rango limitado de matices de color [105]. De esta observación se deriva el que deberíamos esperar una agrupación del color de los píxeles pertenecientes a la piel en cualquier espacio de color en mayor o menor grado (ver figura 2.2). Por tanto, una vez conseguidos unos descriptores de color constantes podremos construir un modelo estadístico del color de la piel y emplearlo en su segmentación.

Dada una secuencia de imágenes en color, construir un seguidor de la cara es fácil si tenemos un buen modelo de la distribución de colores en la imagen. Si I_{rgb} representa los descriptores de color correspondientes a los valores $[RGB]$ de la imagen I, $p(I_{rgb}|piel)$ y $p(I_{rgb}|fondo)$ representan, respectivamente, las funciones de densidad de probabilidad condicionadas del color de la piel y del fondo (asumimos que todo lo que no es piel es fondo), entonces empleando la fórmula de Bayes, la probabilidad de que un píxel de color I_{rgb} sea *piel*, $P(piel|I_{rgb})$, se puede calcular como:

$$P(piel|I_{rgb}) = \frac{p(I_{rgb}|piel)P_p}{p(I_{rgb}|piel)P_p + p(I_{rgb}|fondo)P_f},$$

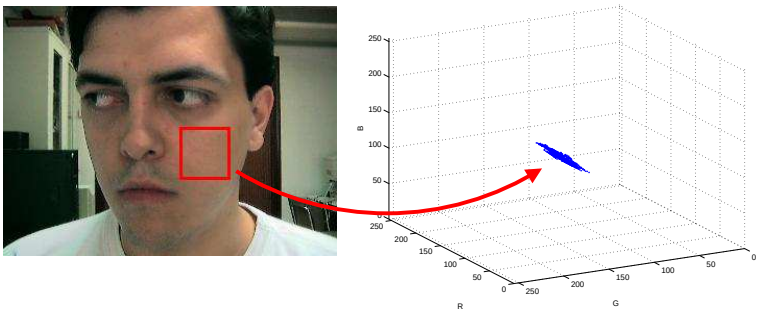


Figura 2.2: Los valores de los píxeles pertenecientes a la piel se agrupan en el espacio RGB.

donde P_p y P_f son las probabilidades a priori de la *piel* y el *fondo*. La transformación $\mathcal{T}(I_{rgb}) = 255 \times P(piel|I_{rgb})$ da como resultado una imagen con valores de gris que representan la probabilidad de ser piel (ver figura 2.3). Sobre esta imagen se puede localizar el rostro empleando un algoritmo de *seguimiento de la moda* (*mode seeking*) [17], mediante el cálculo de la posición y orientación la mancha de píxeles de la cara en cada imagen [12].

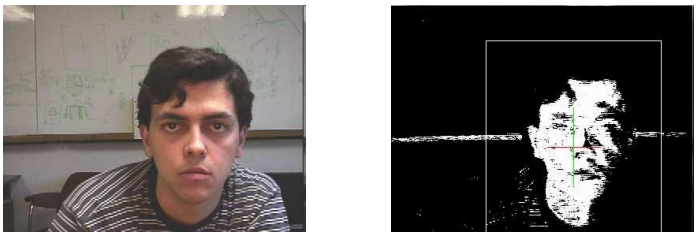


Figura 2.3: Segmentación de la cara basado en el color de la piel. A la izquierda se muestra la imagen de partida, a la derecha la imagen con las probabilidades.

El problema ahora consiste en hacer que el modelo estadístico anterior sea robusto a las variaciones en la iluminación de la escena. Trabajando en alguno de los espacios de color normalizados que proveen *constancia del color* se puede lograr esta meta hasta cierto punto. Y decimos hasta cierto punto, debido a que los descriptores de color que se obtienen no son del todo invariantes a los cambios de iluminación.

2.4. Modelos del color de la piel

Como vimos en la sección 2.3, dada la distribución de probabilidad de la piel y el fondo podemos clasificar los píxeles de una imagen de acuerdo con su probabilidad de pertenencia a la cara. En esta sección veremos que en la literatura se han empleado diferentes formas de representar la piel en el espacio de color elegido.

Algunas aproximaciones simplifican el problema y establecen que el color de la piel se encuentra en una región fija del espacio cromático. Se prefiere un modelo

rápido de calcular que acepte distintas razas e iluminaciones, aunque eventualmente también el fondo, y por tanto no sea muy preciso. Estos modelos se suelen emplear para dirigir la atención de otro método de seguimiento más sofisticado y que suele ser mucho más lento. Kentaro Toyama [93] utiliza directamente el espacio de color RGB y establece como color de la piel a aquel que se encuentra dentro de la región piramidal definida por

$$k_{rg}^{-} < r/g < k_{rg}^{+}$$
$$k_{rb}^{-} < r/b < k_{rb}^{+},$$

la cual acepta todos los colores de todas las razas bajo luz blanca o semejante. Sobottka y Pitas [79], por otro lado, eligen una región del espacio cromático, HSV, siendo $S_{min} = 0,23$, $S_{max} = 0,68$ y $H_{min} = 0^o$,$H_{max} = 50^o$, equivalente a tomar un sector del hexágono HSV entre el rojo y el amarillo. Aunque con ellos se obtienen algoritmos rápidos, con los modelos fijos siempre hay que elegir entre dos extremos, o ser muy tolerante (muchas partes del fondo se clasifican como piel) o ser muy restrictivo y ante iluminaciones diferentes de las que supone el modelo se deja de reconocer el color de la piel (se clasifica incorrectamente).

Yang [59], postula que la distribución de probabilidad del color de la piel en el espacio cromático rg es una normal bivariante para una persona en concreto y una iluminación particular. Por su parte Rasmussen [77], asume que en el espacio de color RGB el color de la piel se encontrará en un elipsoide 3D, con lo que implícitamente utiliza una distribución "cercana" a la normal.

En condiciones reales es muy difícil que tengamos una iluminación completamente uniforme en todas las zonas de la piel. Parece más adecuado suponer que la distribución de probabilidad es una mezcla de normales

$$p(x|piel) = \sum_i p(x|piel, ilumination_i)p(ilumination_i)$$

donde cada normal se corresponde con una de las iluminaciones o zonas de la piel con tonalidad diferente.

Bergasa [7] supone que la distribución de probabilidad de los colores de la imagen completa, incluyendo el fondo, es una mezcla de normales de la que el color de la piel es una componente más. Por tanto se toma como modelo del color de la piel a una única distribución normal de acuerdo con Yang [59], suponiendo entonces que la iluminación sobre la piel es uniforme. En este caso hay que encontrar la gaussiana que corresponde al color de la piel humana y sus parámetros están sujetos a cambios con la iluminación. Mackenna [104], por el contrario, considera que la distribución del color de la piel es una mezcla de normales en el espacio de color HS y ajusta el número de ellas y sus parámetros en un paso previo al seguimiento. El introducir un conjunto de gaussianas debería permitir explicar los datos de una forma más precisa incluso cuando aparezcan zonas de la piel con iluminación diferente. Otras aproximaciones, sin embargo, modelizan la distribución de probabilidad mediante un histograma [12, 68]. Supongamos que la distribución de probabilidad es una normal bivariante

en el espacio cromático rg con media y matriz de covarianza fijas. Debido a que en este espacio de color sólo se elimina la influencia de la geometría de la iluminación, tendremos errores de clasificación cuando el color de la luz sea muy diferente a aquella con la que se estimaron los parámetros de la distribución de probabilidad (ver figura 2.4). Por el contrario, si se calcula el modelo estadístico con el mayor rango de iluminaciones posible tendremos una gran varianza, esto es, tolerancia del modelo funcionando bien con más tipos de iluminación pero clasificando incorrectamente partes del fondo.

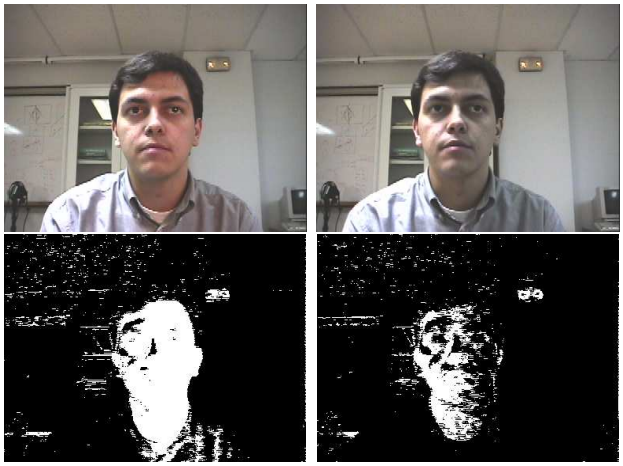


Figura 2.4: Segmentación en el espacio RGB normalizado y suponiendo una distribución normal del color de la piel con parámetros fijos, la clase no piel se modela como una distribución uniforme. A la izquierda podemos ver el resultado sobre una imagen tomada con la misma iluminación que se utilizó para estimar los parámetros, a la derecha con una iluminación muy diferente. En la imagen de la derecha hemos perdido la mitad de la cara.

Existen dos filosofías con las que se suelen manejar los errores en la constancia del color proporcionada por los descriptores de color empleados: tolerancia, delegando el reconocimiento de falsos positivos en otros algoritmos, o adaptación. Bajo la suposición de que las condiciones de visibilidad cambian gradualmente con el tiempo, los modelos de color estadísticos se pueden adaptar para reflejar el cambio del color aparente del objeto seguido [3] (o del fondo de la escena sobre el que se sigue al objeto) [59, 104, 68].

Los distintos métodos de adaptación diferirán en el peso que dan a los datos provenientes de la imagen actual y a la historia pasada del modelo de color. De esta forma podemos distinguir entre *adaptación parcial*, cuando se tiene en cuenta además de una muestra de la imagen, a la historia pasada de las estimaciones de los estadísticos del modelo [59, 104, 7], y *adaptación total* en la que se estiman

[3] En los ambientes en los que existan sombras muy pronunciadas o focos de luz direccionales esta suposición carece de validez.

los nuevos parámetros únicamente con una muestra de color proveniente del objeto seguido en la imagen actual [77]. La adaptación es peligrosa si se produce un fallo de seguimiento ya que en ese caso los datos provenientes de la imagen nos pueden llevar a la distribución de probabilidad, por ejemplo, del fondo de la imagen. Para paliar estos problemas parece más adecuada una adaptación parcial que tome en cuenta la historia pasada, si bien es cierto que no servirá de mucho si la pérdida del objetivo se mantiene durante un número suficiente de imágenes. En [104] se opta por suspender la adaptación cuando se detecta que la verosimilitud del modelo estadístico para los datos que provienen de una ventana sobre el objeto seguido cae por debajo de un umbral. Sin embargo esto no servirá de mucho si se ha producido un cambio muy brusco en el color de la luz, por ejemplo, porque se ha salido de un edificio o se ha abierto una cortina (ya que los parámetros del modelo estadístico serán completamente diferentes).

Una mejor solución es la adoptada en [68]. Todas las posibles cromaticidades de la piel se en encuentran en una región del espacio cromático rg con forma de media luna. Obtienen esa región para la cámara que emplean en el seguimiento. Se actualiza el histograma de la piel con los píxeles pertenecientes a la media luna que se encuentran en la ventana de seguimiento. El conocimiento de los posibles colores que puede presentar la piel bajo diferentes condiciones de iluminación y su uso en el seguimiento, evita en gran medida los problemas de la adaptación del modelo a ciegas.

2.5. Conclusiones

Emplear el color como elemento de seguimiento de objetos es fácil cuando el color de la iluminación no va a cambiar. Sin embargo, cuando tenemos que enfrentarnos con cambios en el color de la luz en la escena, se convierte en un problema bastante más difícil. El problema de *la constancia del color*, no es nada fácil de resolver y no existe hasta la fecha ningún método establecido para resolverlo [72].

Un problema añadido con el color como información visual es que si existen objetos con un color parecido al del objeto seguido, no podremos evitar confundirnos con él. La solución a este último problema pasa por disponer de algún otro elemento de seguimiento que nos permita distinguir al objetivo del resto de objetos de la escena.

No obstante, el color es una característica global a toda la imagen, que nos permite encontrar todos los candidatos posibles de una manera muy rápida y que por tanto es muy útil como paso inicial en un sistema de seguimiento.

Capítulo 3

Extensiones a la hipótesis del mundo gris

En este capítulo se presenta nuestra propuesta para resolver el problema de la constancia del color en secuencias de imágenes. Los dos algoritmos que introduciremos se basan en *la hipótesis del mundo gris* (en inglés *Grey World*), en adelante GW, [38] y explotan la información presente en una secuencia de imágenes para calcular los cambios en la iluminación entre dos instantáneas. Ambos algoritmos son procedimientos de normalización del color alternativos a la normalización RGB. Como se mostrará en los experimentos, los nuevos procedimientos de normalización del color son más robustos a cambios bruscos en el color de la luz que la normalización RGB.

3.1. Constancia del color basada en la hipótesis del mundo gris

De acuerdo con el modelo de formación de imágenes en color que vimos en la sección 2.1, dos píxeles $I(ij)$ e $I(kl)$ de una imagen tendrán los valores $[RGB]$ que se muestran a continuación: $[s_{ij}\alpha r_{ij}, s_{ij}\beta g_{ij}, s_{ij}\gamma b_{ij}]$, $[s_{kl}\alpha r_{kl}, s_{kl}\beta g_{kl}, s_{kl}\gamma b_{kl}]$, donde $[r_{ij}, g_{ij}, b_{ij}]$ y $[r_{kl}, g_{kl}, b_{kl}]$ representan la reflectancia de la superficie, esto es, el color real del objeto, independiente de la iluminación. Por otro lado s_{ij} y s_{kl} son los factores dependientes de la geometría de la iluminación correspondientes a cada pixel. Por último, α, β y γ son los valores correspondientes a un cambio en el color de la iluminación y que son iguales para todos los píxeles (ver capítulo 2).

El algoritmo GW propuesto por Buchsbaum [15] asume que la reflectancia media en una imagen con suficientes superficies distintas es gris. Así que la intensidad media reflejada corresponde al color de la iluminación y puede emplearse para calcular los descriptores de color. Este algoritmo fue mejorado en [76] mediante la obtención un modelo medio de la reflectancia de las superficies y proponiendo un procedimiento para calcular la reflectancia media de la imagen. En base a lo anterior, la normalización del color propuesta en el algoritmo GW consiste en dividir cada canal

de color por su media:

$$[s_{ij}\alpha r_{ij}, s_{ij}\beta g_{ij}, s_{ij}\gamma b_{ij}] \mapsto \left[\frac{\alpha s_{ij} r_{ij}}{\frac{\alpha}{n}\sum_{\forall ij\in I} s_{ij} r_{ij}}, \frac{\beta s_{ij} g_{ij}}{\frac{\beta}{n}\sum_{\forall ij\in I} s_{ij} g_{ij}}, \frac{\gamma s_{ij} b_{ij}}{\frac{\gamma}{n}\sum_{\forall ij\in I} s_{ij} b_{ij}}\right] \quad (3.1)$$

Llamaremos *normalización GW básica* a la ecuación (3.1). Es invariante a las variaciones del color de la iluminación (α, β, γ), pero asume que s_{ij} es constante lo que constituye una limitación importante debido a que:

- No tiene en cuenta todas las variaciones de la intensidad de la iluminación (algunas de estas variaciones se cancelarán con los coeficientes de color).
- Falla cuando las reflectancias de las superficies (r_{ij},g_{ij},b_{ij}) cambian, p.ej. cuando un objeto aparece o desaparece de la escena. Lo que quiere decir que GW sólo es válido para escenas estáticas. En la siguiente sección introduciremos una extensión dinámica al algoritmo GW que soluciona este problema empleando la información redundante disponible en una secuencia de imágenes.

Vamos a definir la *reflectancia geométrica media de una imagen*, $\bar{\mu}$, como

$$\bar{\mu} = [\mu_r, \mu_g, \mu_b] = \left[\frac{1}{n}\sum_{ij\in I} s_{ij} r_{ij}, \frac{1}{n}\sum_{ij\in I} s_{ij} g_{ij}, \frac{1}{n}\sum_{ij\in I} s_{ij} b_{ij}\right],$$

donde n es el número de píxeles en la imagen. Representa los valores $[RGB]$ medios una vez que el componente del color de la iluminación ha sido eliminado.

Si asumimos que la reflectancia geométrica media es constante en toda la secuencia, entonces la siguiente normalización, a la que llamaremos *Normalización GW Proyectiva*, elimina los cambios en la intensidad y en el color de la iluminación:

$$\begin{aligned}[s_{ij}\alpha r_{ij}, s_{ij}\beta g_{ij}, s_{ij}\gamma b_{ij}] &\mapsto \left[\frac{s_{ij} r_{ij}}{\mu_r}, \frac{s_{ij} g_{ij}}{\mu_g}, \frac{s_{ij} b_{ij}}{\mu_b}\right] \mapsto \\ &\mapsto \left[\frac{r_{ij}}{(\frac{r_{ij}}{\mu_r} + \frac{g_{ij}}{\mu_g} + \frac{b_{ij}}{\mu_b})}, \frac{g_{ij}}{(\frac{r_{ij}}{\mu_r} + \frac{g_{ij}}{\mu_g} + \frac{b_{ij}}{\mu_b})}\right]\end{aligned} \quad (3.2)$$

Esta nueva normalización es válida únicamente en escenas estáticas. En la siguiente sección presentaremos una extensión dinámica a la normalización GW básica para solucionar este problema.

3.2. Seguimiento del rostro con GW Dinámico

En la literatura, como vimos en la sección 2.4, aparecen diferentes formas de modelar las distribuciones de color. Mackenna [104] emplea modelos basados en una mezcla de normales, mientras que Schwerdt [57] y Martinkauppi [68] utilizan representaciones basadas en histogramas. En nuestros experimentos hemos llegado

a la conclusión de que los modelos basados en histogramas son más rápidos en ejecución y permiten representar fielmente la distribución de los datos, siempre que se disponga de suficientes muestras para construirlo. Por otra parte, los modelos paramétricos son adecuados cuando las muestras se encuentran más dispersas.

Ya que la normalización GW básica define un espacio de color de dimensión tres, $I_{\hat{r}\hat{g}\hat{b}}$, y dado el pequeño número de muestras disponibles en una imagen, proponemos modelar la distribución de los descriptores de la piel de la normalización GW básica con un modelo paramétrico: una normal. Como se puede observar en la figura 3.1, $p(I_{\hat{r}\hat{g}\hat{b}}|piel)$ es aproximadamente normal. A la derecha se muestran la gráfica de la χ^2 de la distribución de $I_{\hat{r}\hat{g}\hat{b}}$ y las gráficas de la normal de las distribuciones marginales de $I_{\hat{r}}$, $I_{\hat{g}}$ y $I_{\hat{b}}$. Mediante el análisis de estas gráficas podemos verificar que la hipótesis $p(I_{\hat{r}\hat{g}\hat{b}}|piel) \sim N(\bar{m}_s, \Sigma_s, I_{\hat{r}\hat{g}\hat{b}})$ es plausible. Por otro lado, no es posible encontrar un modelo analítico para el *fondo*, así que lo modelaremos como una distribución de probabilidad uniforme, $u_b(I_{\hat{r}\hat{g}\hat{b}})$.

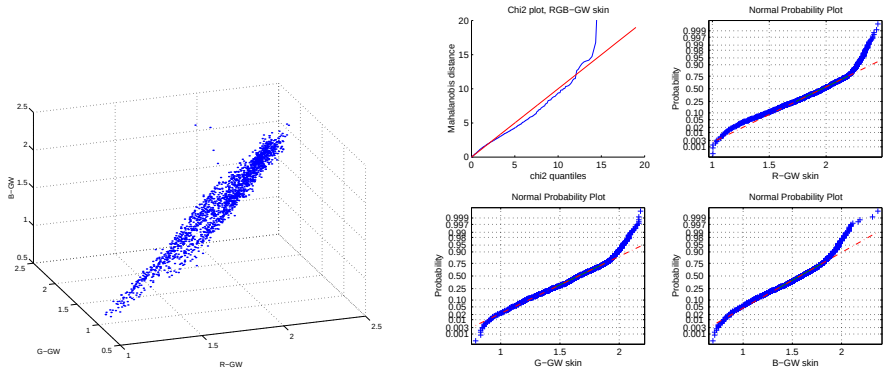


Figura 3.1: Distribución de probabilidad de los descriptores de la normalización GW básica para el color de la piel. A la izquierda se muestra el agrupamiento formado por los descriptores del color de la piel. A la derecha se muestra la gráfica de la χ^2 para la distribución multivariante y la gráficas de la normal para las marginales.

Si aproximamos las distribuciones a priori $P_s \approx n_s/n$ y $P_b \approx n_b/n$, donde n_s y n_b son respectivamente el número de píxeles de la *piel* y del *fondo*, entonces

$$P(piel|I_{rgb}) = \frac{n_s\, N(\bar{m},\, \Sigma, I_{\hat{r}\hat{g}\hat{b}})}{n_s\, N(\bar{m},\, \Sigma, I_{\hat{r}\hat{g}\hat{b}}) + n_b\, u_b(I_{\hat{r}\hat{g}\hat{b}})}.$$

Por otro lado, la normalización GW proyectiva define un espacio de color bidimensional, $I_{\hat{r}\hat{g}}$. Al ser menor la dimensión, aumenta la densidad de puntos y podemos modelar las distribuciones de color como histogramas, tanto para la piel, $h_s(I_{\hat{r}\hat{g}})$, como para el fondo, $h_b(I_{\hat{r}\hat{g}})$. El resultado es que en nuestra implementación modelamos la distribución a posteriori como:

$$P(piel|I_{rgb}) = \frac{h_s(I_{\hat{r}\hat{g}})}{h_s(I_{\hat{r}\hat{g}}) + h_b(I_{\hat{r}\hat{g}})}.$$

3.2.1. El algoritmo GW dinámico

El principal problema del algoritmo GW es que se concibió para imágenes estáticas y falla cuando se produce un gran cambio en la reflectancia geométrica media de la imagen. En esta sección proponemos una extensión dinámica al algoritmo GW (DGW) con la que se detectará esta situación y se actualizará la distribución de probabilidad de los descriptores de GW.

En lo que sigue asumiremos que existe una partición de la secuencia de imágenes en un conjunto de subsecuencias. La partición es de tal forma que la reflectancia geométrica media es constante en el interior de cada subsecuencia. La normalización GW básica (igualmente la normalización GW proyectiva), por tanto, se puede emplear como criterio de constancia del color en cada una de ellas. Usaremos la primera imagen de cada subsecuencia como *imagen de referencia*. El resto de imágenes de la subsecuencia se segmentarán utilizando los descriptores de color de la imagen de referencia.

Sean I^r_{rgb}, I^t_{rgb} y I^{t-1}_{rgb}, la imagen de referencia, la imagen actual y la anterior respectivamente, $F^r_{\hat{r}\hat{g}\hat{b}}$ los píxeles de la cara en el espacio de los descriptores GW, $\bar{\mu}^{I^t}_{rgb}$ el valor medio de cada canal de color en I^t_{rgb}, $\bar{\mu}^{F^{t-1}}_{\hat{r}\hat{g}\hat{b}}$ y $\bar{\mu}^{F^t}_{\hat{r}\hat{g}\hat{b}}$ la media de los descriptores GW para los píxeles de la cara en la imagen anterior y en la imagen actual respectivamente, y $\mathcal{E}$ la distribución estadística de los descriptores de color GW para la imagen de referencia.

La extensión dinámica al algoritmo GW básico se basa en el hecho de que cuando se detecta un cambio en la reflectancia geométrica media ($\bar{\mu}$), los descriptores de color GW para la imagen actual $I^t_{\hat{r}\hat{g}\hat{b}}$ se pueden calcular todavía con las medias de la imagen anterior, $\bar{\mu}^{I^{t-1}}_{rgb}$. En este momento se segmenta la imagen actual con los valores medios de los píxeles de la anterior, y la imagen actual se convierte en la nueva imagen de referencia.

El problema ahora consiste en cómo detectar un cambio de subsecuencia. Lo haremos detectando un cambio en la reflectancia geométrica media. No lo podremos hacer analizando $\bar{\mu}^I_{rgb}$, ya que también cambia con el color de la iluminación. La solución viene de la monitorización de la media de los descriptores GW en los píxeles de la cara. Ya que son invariantes a los cambios del color de la iluminación, la causa de un cambio en estos descriptores es necesariamente de un cambio en la reflectancia geométrica media. Como veremos en los experimentos, si $n^{F^t} = ||\bar{\mu}^{F^t}_{\bar{r}\bar{g}\bar{b}} - \bar{\mu}^{F^{t-1}}_{\bar{r}\bar{g}\bar{b}}||$ es una aproximación a la norma de la derivada de los descriptores medios de la cara en la imagen t, entonces podremos detectar el cambio de subsecuencia de GW mediante un valor demasiado grande de n^F en la imagen t.

En base a estas ideas proponemos el algoritmo GW Dinámico (DGW) (ver algoritmo 3.1). El punto débil del algoritmo DGW consistirá en un cambio simultáneo de la geometría de la iluminación y la reflectancia geométrica media.

/* Iniciar el modelo de la imagen de referencia con segmentación basada en el movimiento y un modelo de color precalculado */
$[\mathcal{E}, \bar{\mu}^{F^r}_{\hat{r}\hat{g}\hat{b}}]$ = InitTracking()

/* bucle del seguidor */
while *true* **do**
  /* Media de los valores rgb */
  $\bar{\mu}^{I^t}_{rgb} = \text{Mean}(I^t_{rgb})$
  /* Normalización GW */
  $I^t_{\hat{r}\hat{g}\hat{b}} = \frac{I^t_{rgb}}{\bar{\mu}^{I^t}_{rgb}}$
  /* Segmentar imagen */
  $F^t_{\hat{r}\hat{g}\hat{b}} = \text{ProbabilisticSegment}(I^t_{\hat{r}\hat{g}\hat{b}}, \mathcal{E})$
  /* Calcular descriptores GW medios de la cara */
  $\bar{\mu}^{F^t}_{\hat{r}\hat{g}\hat{b}} = \text{ComputeAvgFaceGW}(F^t_{\hat{r}\hat{g}\hat{b}})$
  $n^{F^t} = ||\bar{\mu}^{F^t}_{\bar{r}\bar{g}\bar{b}} - \bar{\mu}^{F^{t-1}}_{\bar{r}\bar{g}\bar{b}}||$
  if $n^{F^t} > \Delta$ **then**
    /* Cambio de subsecuencia */
    /* Normalización GW con la media anterior */
    $I^t_{\hat{r}\hat{g}\hat{b}} = \frac{I^t_{rgb}}{\bar{\mu}^{I^{t-1}}_{rgb}}$
    /* Segmentar imagen */
    $F^t_{\hat{r}\hat{g}\hat{b}} = \text{ProbabilisticSegment}(I^t_{\hat{r}\hat{g}\hat{b}}, \mathcal{E})$
    /* Actualizar imagen de referencia */
    $I^r_{\hat{r}\hat{g}\hat{b}} = I^t_{\hat{r}\hat{g}\hat{b}}$
    /* Descriptores GW de la cara */
    $\bar{\mu}^{F^r}_{\hat{r}\hat{g}\hat{b}} = \text{ComputeAvgFaceGW}(F^t_{\hat{r}\hat{g}\hat{b}})$
    /* Distribución de color de referencia */
    $[\mathcal{E}] = \text{ColourDistrib}(F^t_{\hat{r}\hat{g}\hat{b}})$
  end
end

Algoritmo 3.1: Algoritmo GW dinámico.

3.3. Experimentos

En esta sección pretendemos validar las hipótesis expuestas en las secciones anteriores. Realizaremos los experimentos con tres secuencias con fondos diferentes y con condiciones de iluminación cambiantes (a través de fluorescentes en el techo, luz azulada, y una lámpara de tungsteno de intensidad variable, luz rojiza). En todos los experimentos se ha empleado una distribución de probabilidad normal como modelo del color de la piel en el caso de la normalización GW básica y un histograma para el caso de la normalización GW proyectiva. Para el caso de la normalización RGB también se ha utilizado un histograma. En todos los casos el modelo de color de la piel se construye sobre la primera imagen segmentando la piel manualmente.

En todos los experimentos hemos establecido el umbral para el cambio de subsecuencia en un valor de n^F por encima de 0,08. También en todos los experimentos se considerarán píxeles de la piel aquellos que después de la clasificación (en la imagen segmentada) superen la probabilidad de 0,33.

En el primer experimento confirmaremos la hipótesis sobre la que descansa el algoritmo DGW: las variaciones en la reflectancia geométrica media se pueden detectar, y la imagen de referencia de cada subsecuencia se puede segmentar. Utilizaremos una secuencia de 199 imágenes con un objeto verde apareciendo en un determinado momento y con variaciones en la geometría de la iluminación en diferentes instantes. El resultado de este experimento se puede observar en las figuras 3.2 y 3.3. En la figura 3.2 se muestran cuatro imágenes de la secuencia, cada una en una columna de la figura, representando respectivamente, la primera imagen de la secuencia (imagen 1), un cambio en la iluminación apagándose las luces del techo (imagen 26), y la aparición y desaparición de un objeto (imágenes 88 y 139). En este experimento el algoritmo detecta tres subsecuencias (de la 1 a la 87, de la 88 a la 138 y de la 139 a la 199). Esto último es claramente visible en la gráfica de los descriptores medios de la cara en la figura 3.3. En la imagen 26 se apaga la luz fluorescente del techo. Esta variación en la geometría de la iluminación se puede detectar en la gráfica de n^F con la aparición de valores extremos en la misma. En este caso la segmentación es buena a pesar de que estamos ante una prueba del "peor caso". En situaciones similares con variaciones más fuertes en la geometría de la iluminación, el algoritmo puede no ser capaz de segmentar la imagen y eventualmente puede perderse. Las imágenes 88 y 139 muestran la primera imagen segmentada de cada una de las dos últimas subsecuencias, y coinciden con la aparición y desaparición de un objeto en la escena. Lo que podemos observar aquí es cómo el sistema detecta un cambio de subsecuencia y segmenta correctamente las imágenes.

El propósito del siguiente experimento es verificar que la extensión dinámica al algoritmo GW es necesaria. Queremos probar qué pasaría si segmentamos la secuencia del primer experimento con la normalización GW sin la extensión dinámica. En la figura 3.4 se muestran los resultados de la normalización GW básica, sin y con actualización del modelo de color. El modelo de color inicial se invalida en cuanto ocurre un cambio en la reflectancia geométrica media. El modelo inicial vuelve a ser válido gradualmente en cuanto el objeto comienza a desaparecer.

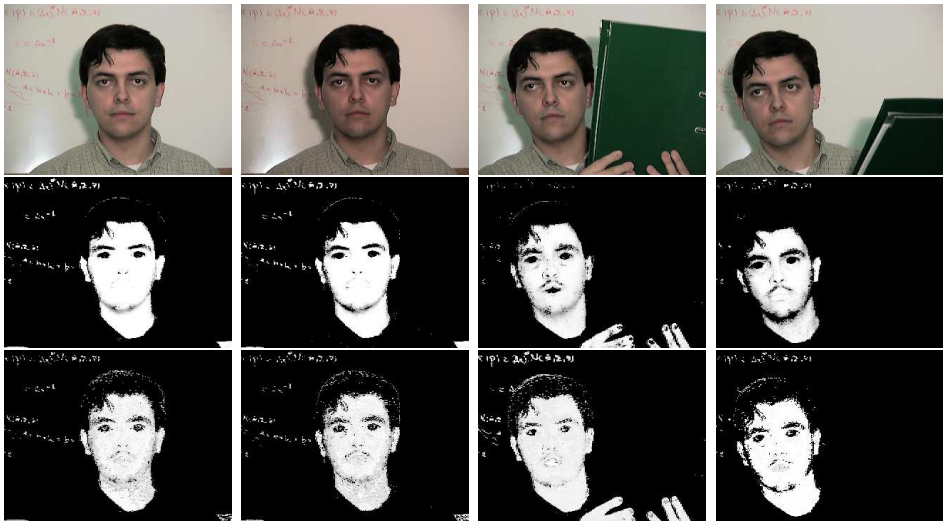


Figura 3.2: Primer experimento, detección del cambio de subsecuencia. En la primera fila se muestran las imágenes 1, 26, 88 y 139 de una secuencia de 199. En la segunda las imágenes segmentadas mediante la normalización GW básica con extensión dinámica y en la tercera fila aparecen las imágenes segmentadas con la normalización GW proyectiva también con la extensión dinámica.

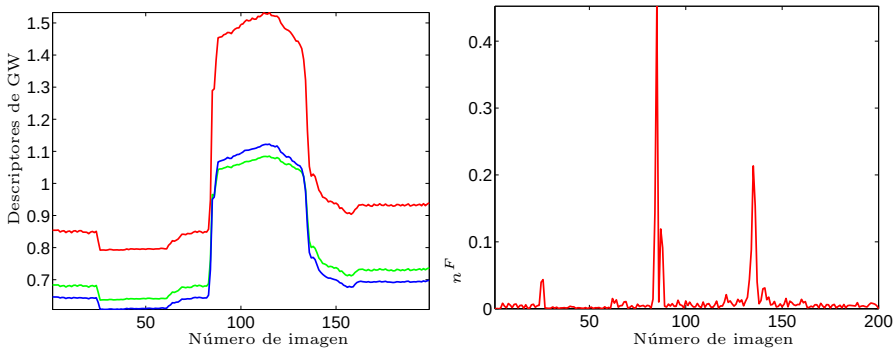


Figura 3.3: Gráficas de los descriptores de GW medios (izquierda) - en rojo aparece la gráfica para el descriptor GW del canal R de cada imagen, en verde el de la G, y en azul el de la B -, y de n^F (ver texto) para el primer experimento.

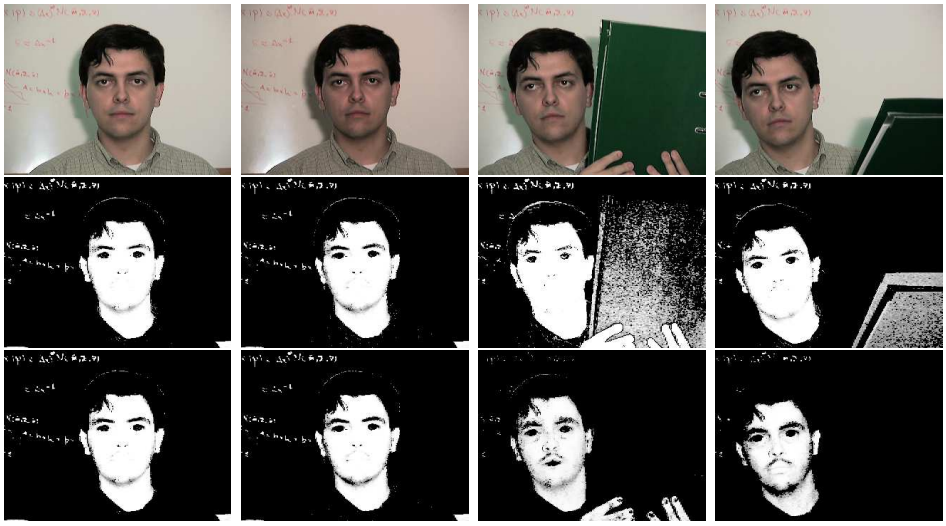


Figura 3.4: Comprobación de la necesidad de la extensión dinámica del algoritmo GW. En la primera fila imágenes 1, 26, 88 y 139 de una secuencia de 199. En la segunda fila, imágenes segmentadas mediante la normalización GW básica sin extensión dinámica. En la tercera fila aparecen las imágenes segmentadas con la normalización GW básica con la extensión dinámica propuesta.

En el tercer experimento se compara el comportamiento del algoritmo DGW con la normalización RGB como algoritmo de constancia del color. Se emplea una secuencia, de 199 imágenes, en la que el fondo es más complicado que en el primer experimento (libros y estanterías). En la figura 3.5 se pueden observar en cada columna una de las imágenes de la secuencia que representan las situaciones más significativas (ver descriptores GW medios de la cara en la figura 3.6): la imagen inicial, apagado de una luz de tungsteno frontal a la persona (imagen 40) encendido parcial de la luz de tungsteno frontal (imagen 75) y aparición de un objeto verde (imagen 110 a 145). En la primera fila y en la quinta filas se muestran las imágenes originales, y debajo de ellas aparecen las respectivas imágenes segmentadas. Se muestran inmediatamente debajo de las imágenes originales los resultados de la segmentación con la normalización RGB, a continuación con la normalización GW básica y por último la normalización GW proyectiva. De la inspección visual de estos resultados podemos concluir que la normalización GW tanto básica como proyectiva arrojan resultados similares, aunque la proyectiva parece levemente más robusta ante cambios en la intensidad de la iluminación. Los resultados de la segunda fila representan un mejor comportamiento del algoritmo DGW frente a la normalización RGB cuando se produce un cambio muy brusco del color de la iluminación.

En el último experimento empleamos una secuencia "difícil" dado que aparecen en el fondo superficies del mismo color que la piel y cambios importantes en el color de la iluminación. En esta secuencia (ver figura 3.7) comenzamos iluminando con una luz de tungsteno y luces fluorescentes en el techo, se apaga la luz de tungsteno

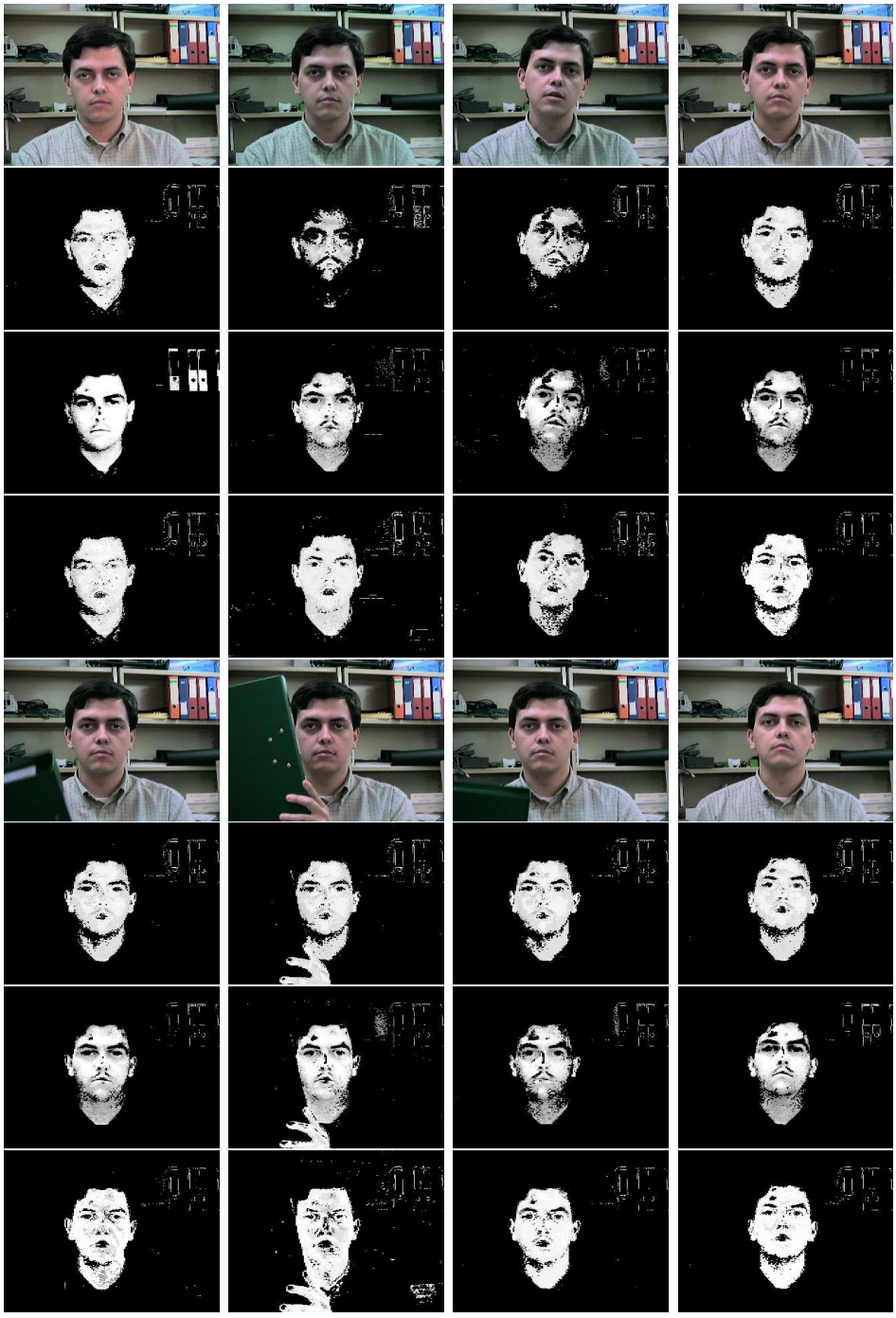


Figura 3.5: Tercer experimento: comparación del algoritmo GW dinámico con la normalización RGB. En la primera fila aparecen las imágenes 1, 50, 75 y 100 de una secuencia de 199. Las tres siguientes filas presentan los resultados para las imágenes anteriores aplicando la normalización RGB (fila 2), la normalización GW básica (fila 3) y la normalización GW proyectiva (fila 4). En la quinta fila se muestran las imágenes 111, 121, 140 y 188. Las tres siguientes filas presentan los resultados para las imágenes de la quinta fila de la normalización RGB (fila 6), normalización GW básica (fila 7) y normalización GW proyectiva (fila 8).

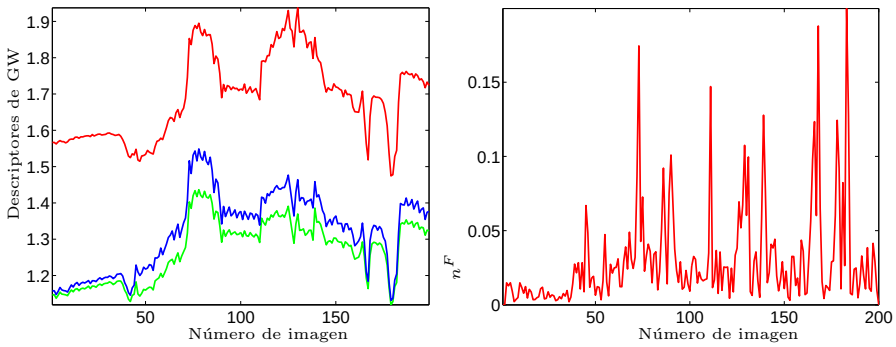


Figura 3.6: Gráficas de los descriptores de GW medios (izquierda) - en rojo aparece la gráfica para el descriptor GW del canal R de cada imagen, en verde el de la G, y en azul el de la B -, y de n^F (ver texto) para el segundo experimento.

(imágenes 18 a 65), se enciende la de tungsteno y se apagan los fluorescentes (imágenes 87 a 90), se encienden las luces del techo y finalmente aparece un objeto de color azul (imágenes 117 a 137). Claramente todos estos eventos tienen su reflejo en los descriptores de GW medios de la cara en la figura 3.8. Si observamos los resultados en la figura 3.7, podemos confirmar que la normalización RGB falla cuando el color de la iluminación cambia bruscamente mientras que las dos versiones basadas en GW no tienen problemas en ese caso (ver resultados para las imágenes 16 y 25).

3.4. Conclusiones

En este capítulo hemos presentado dos algoritmos de normalización de color alternativos al RGB normalizado: la normalización GW básica y la proyectiva. Al tratarse de procedimientos de normalización del color se les puede aplicar cualquier estrategia de adaptación del modelo de color.

El algoritmo GW se concibió para trabajar sobre imágenes estáticas, sin embargo, nosotros hemos desarrollado la extensión dinámica a GW que se puede emplear tanto en la normalización GW básica como en la proyectiva. Estas extensiones a GW se han diseñado para trabajar en tiempo real en secuencias de imágenes con condiciones de trabajo cambiantes. En los experimentos presentados en la sección anterior, la extensión dinámica, tanto con la normalización GW básica como con la proyectiva, se comportó mejor que la normalización RGB cuando se produce un cambio brusco en el color de la iluminación. La normalización GW proyectiva es preferible a la básica porque se comporta mejor con las variaciones de la intensidad de la iluminación y porque trabaja en un espacio de color bidimensional. Por otro, lado el caso menos favorable para nuestro algoritmo ocurre cuando se produce un cambio en la geometría de la iluminación junto con un cambio en la reflectancia geométrica media.

Un seguidor basado en información visual de bajo nivel como es el color, presen-

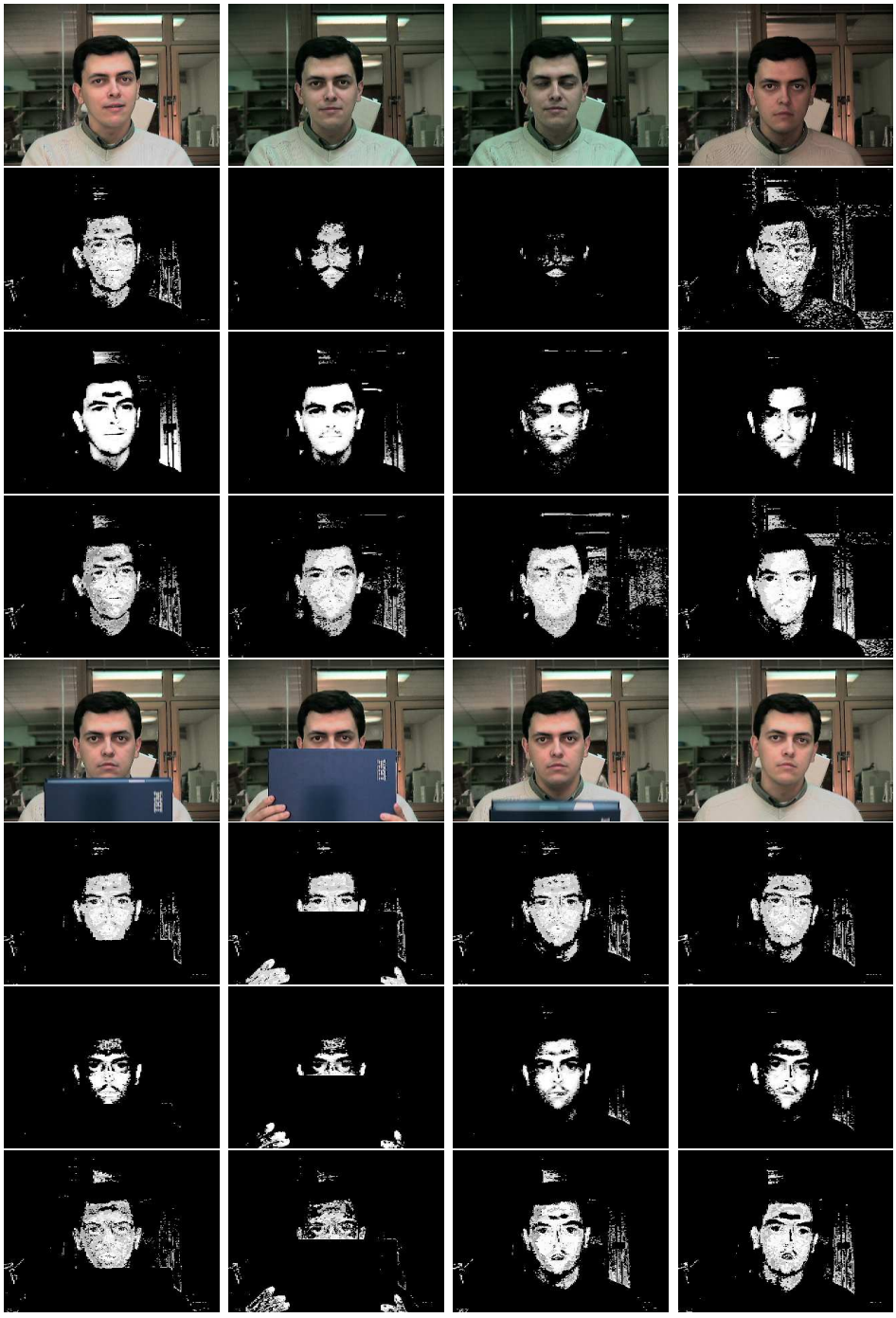


Figura 3.7: Cuarto experimento: comparación del algoritmo GW dinámico con la normalización RGB en una secuencia con grandes cambios en el color de la iluminación. En la primera fila aparecen las imágenes 1, 16, 25 y 88 de una secuencia de 199. Las tres siguientes filas presentan los resultados para las imágenes anteriores aplicando la normalización RGB (fila 2), la normalización GW básica (fila 3) y la normalización GW proyectiva (fila 4). En la quinta fila se muestran las imágenes 118, 128, 136 y 159. Las tres siguientes filas presentan los resultados para las imágenes de la quinta fila de la normalización RGB (fila 6), normalización GW básica (fila 7) y normalización GW proyectiva (fila 8).

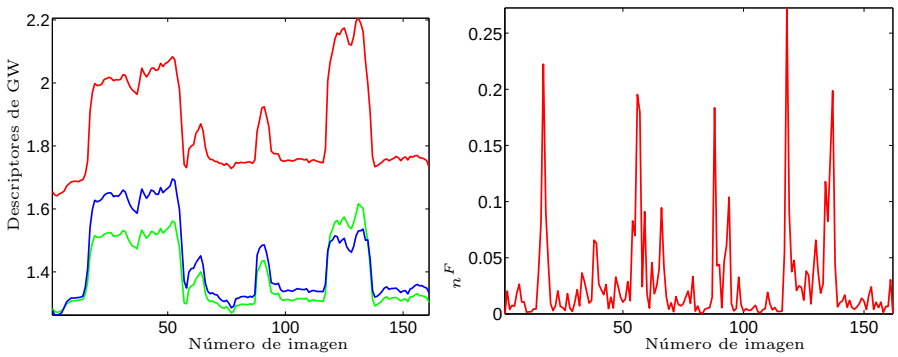


Figura 3.8: Gráficas de los descriptores de GW medios (izquierda) - en rojo aparece la gráfica para el descriptor GW del canal R de cada imagen, en verde el de la G, y en azul el de la B -, y de n^F (ver texto) para el último experimento.

tará siempre situaciones de fallo. En esta parte de la tesis se han analizado algunos de los puntos débiles del algoritmo de normalización RGB. La normalización DGW tampoco es perfecta, ya que su comportamiento se puede ver muy afectado por cambios simultáneos en la geometría de la iluminación y en la reflectancia geométrica media, sin embargo, esta situación es más difícil de observar que un simple cambio en la iluminación, que constituye la condición de fallo de la normalización RGB.

El seguimiento basado en el color, en general, tiene dificultades con los objetos con color similar al buscado en el fondo. Pese a todo, los seguidores basados en el color son buenos como procedimiento de estimación inicial o verificación de la posición de la cara en el plano imagen, o como procedimiento de recuperación cuando un algoritmo de seguimiento más preciso, pero lento, no es capaz de estimar el movimiento de la cara.

Parte III

La textura de la cara como elemento de seguimiento

Capítulo 4

Seguimiento de regiones planas con niveles de gris

El seguimiento de regiones planas es un tema de interés en el ámbito de la Visión por Computador con aplicaciones, entre otras, en realidad aumentada [82], navegación de robots móviles [95], seguimiento del rostro [9], o la generación de imágenes con super-resolución [30]. Dado que la cara se puede considerar plana en posiciones cercanas a la frontal a la cámara, emplearemos un seguidor de planos para localizar el rostro en la imagen. Este tipo de técnicas nos permitirán una mayor precisión en su localización que el seguimiento basado en el color de la piel.

Una de las aproximaciones más usuales al seguimiento de regiones planas es la alineación incremental. Consiste en definir un modelo de movimiento y minimizar las diferencias en los niveles de gris entre dos imágenes a través de una función de coste no lineal. En la literatura se han propuesto diferentes aproximaciones a este problema de optimización que difieren en el procedimiento de minimización, en el modo de actualizar los parámetros del modelo de movimiento (suma o composición), y en el término de la función de coste en que se realiza la aproximación lineal (cuando se minimiza la función de coste usando el método de Gauss-Newton).

En este capítulo explicaremos en qué consiste la alineación incremental de imágenes y expondremos las principales aproximaciones al problema aparecidas en la literatura.

4.1. Alineación con movimiento rígido

Sea $\bar{x}$ la posición de un punto en una imagen e $I(\bar{x}, t)$ el nivel de gris de esa posición en la imagen obtenida en el instante t. Sea $\mathcal{R} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N\}$ un conjunto de N puntos de la imagen del objeto a seguir (*región objetivo*), cuyos niveles de gris son conocidos en una imagen de referencia $I_r(\bar{x})$. Estos puntos junto con sus valores de gris en la imagen de referencia constituyen la *plantilla de referencia* a seguir.

El movimiento relativo entre el objeto en seguimiento y la cámara provoca que la región objetivo se desplace y deforme en la imagen. Vamos a representar esta deformación mediante un *modelo de movimiento* $f(\bar{x}, \bar{\mu})$ parametrizado mediante

$\bar{\mu} = (\mu_1, \mu_2, \ldots, \mu_n)^\top$, con $N > n$ y f derivable tanto en $\bar{x}$ como en $\bar{\mu}$. El vector $\bar{\mu}_I$ representará el conjunto de parámetros del movimiento nulo o transformación identidad, $f(\bar{x}, \bar{\mu}_I) = \bar{x}$, y el vector $\bar{\mu}_0$ representará la posición y deformación de la región objetivo en la imagen inicial de la secuencia, $I_r(\bar{x}) = I(f(\bar{x}, \bar{\mu}_0), t_0)$, y lo supondremos conocido.

Asumiremos que las variaciones en los valores de gris sólo se deben al movimiento, lo que significa que la *ecuación de constancia de los niveles de gris* se verifica para todos los píxeles en $\mathcal{R}$

$$I_r(\bar{x}) = I_r(f(\bar{x}, \bar{\mu}_I)) = I(f(\bar{x}, \bar{\mu}_t^*), t) \;\; \forall \bar{x} \in \mathcal{R}, \tag{4.1}$$

donde $\bar{\mu}_t^*$ es la deformación de la imagen actual en el instante t. Por otro lado, $I(f(\bar{x}, \bar{\mu}), t)$ es la imagen rectificada con los parámetros de movimiento $\bar{\mu}$, para lo que puede ser necesario interpolación en los niveles de gris de la imagen capturada en el instante t, y tiene la forma y tamaño de la plantilla de referencia $I_r(\bar{x})$ (ver figura 4.1).

En este caso, el seguimiento de un objeto implica estimar el vector de parámetros de movimiento, $\bar{\mu}_t$, en todas las imágenes de la secuencia. La estimación se puede realizar mediante la minimización de la función:

$$\bar{\mu}_t = arg \min_{\bar{\mu}} \| \mathbf{I}(f(\bar{x}, \bar{\mu}), t) - \mathbf{I}_r(\bar{x}) \|^2, \tag{4.2}$$

donde $\mathbf{I}(\bar{x})$ es el vector columna con todos los niveles de gris de la imagen $I(\bar{x})$.

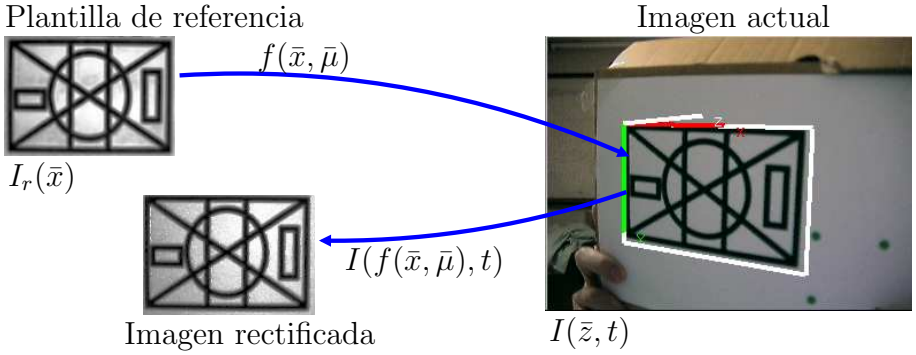


Figura 4.1: Elementos fundamentales en la alineación incremental de imágenes.

En general, la función (4.2) es difícil de minimizar, ya que plantea una relación no lineal entre la variación de los niveles de gris y de los parámetros de movimiento del objeto. En la literatura se ha resuelto tradicionalmente partiendo de una hipótesis de continuidad del movimiento [67, 40, 81, 4]. Si en un instante de tiempo inicial, t_0, los parámetros del modelo de movimiento, $\bar{\mu}_0$, son conocidos, entonces (4.2) se puede linealizar localmente mediante un desarrollo en serie de Taylor alrededor del punto $(\bar{\mu}_0, t_0)$. De esta forma, la posición de la región objetivo en el instante $t_0 + \delta t$ se puede estimar linealmente, siempre que el desplazamiento de la región objetivo en ambos instantes de tiempo, $\delta\bar{\mu}$, sea pequeño. Mediante la repetición de este procedimiento en cada imagen de la secuencia, se pueden estimar los parámetros del modelo $\bar{\mu}_t$ en cualquier instante de tiempo $t > t_0$.

4.1.1. Algoritmo aditivo de Lucas y Kanade

En esta sección describiremos el procedimiento de alineación propuesto por Lucas y Kanade [67], que supuso el punto de partida para muchos otros desarrollos posteriores. Asumiremos que conocemos una estimación de los parámetros del modelo de movimiento en el instante t, lo que significa que tenemos un $\bar{\mu}_t$ tal que $I_r(\bar{x}) \approx I(f(\bar{x}, \bar{\mu}_t), t)$ (ver figura 4.2). El problema de seguimiento se puede describir como la estimación, a partir de $I(\bar{z}, t+\delta t)$ (la imagen en el instante de tiempo $t+\delta t$), del incremento $\delta\bar{\mu}_a$ en los parámetros de movimiento de tal forma que

$$I_r(\bar{x}) \approx I(\bar{z} = f(\bar{x}, \bar{\mu}_{t+\delta t}), t + \delta t) = I(f(\bar{x}, \bar{\mu}_t + \delta\bar{\mu}_a), t + \delta t).$$

Es importante destacar que el incremento en los parámetros del modelo se suman a la estimación previa, lo que significa que $\bar{\mu}_{t+\delta t} = \bar{\mu}_t + \delta\bar{\mu}_a$. A esta forma de actualizar los parámetros de movimiento la llamaremos *aditiva* y de ahí el subíndice a en $\delta\bar{\mu}_a$.

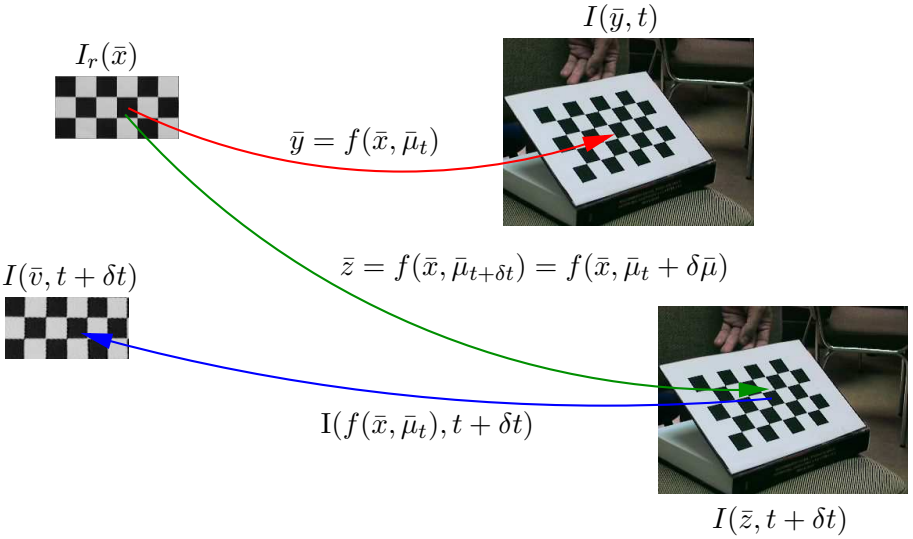


Figura 4.2: Elementos fundamentales en el algoritmo de Lucas y Kanade. $I_r(\bar{x})$ es la imagen de la plantilla de referencia. $I(\bar{y}, t)$ es la imagen capturada en el instante de tiempo t, $I(\bar{z}, t+\delta t)$ es la imagen capturada en el instante de tiempo $t+\delta t$ e $I(\bar{v}, t+\delta t)$ es la imagen rectificada en el instante $t + \delta t$ con los parámetros de movimiento $\bar{\mu}_t$. A través de la función de movimiento f se pueden relacionar las coordenadas de una imagen con las de otra (conociendo los parámetros de movimiento) o se puede rectificar una imagen (sus niveles de gris) sobre las coordenadas de otra (*warping*).

Teniendo en cuenta lo anterior, la minimización que se presentó en (4.2) se puede reescribir como

$$\delta\bar{\mu}_a = arg\min_{\delta\bar{\mu}} \| \mathbf{I}(f(\bar{x}, \bar{\mu}_t + \delta\bar{\mu}), t + \delta t) - \mathbf{I}_r(\bar{x}) \|^2, \tag{4.3}$$

que representa una función de coste no lineal en el vector incremento de parámetros. Se puede linealizar mediante un desarrollo en serie de Taylor alrededor de $\bar{\mu}_t$. Si nos

quedamos con los términos de primer orden tendremos

$$\mathbf{I}(f(\bar{x}, \bar{\mu}_t + \delta\bar{\mu}_a), t + \delta t) \approx \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t + \delta t) + \mathbf{M}(\bar{\mu}_t, t + \delta t)\delta\bar{\mu}_a \tag{4.4}$$

donde

$$\mathbf{M}(\bar{\mu}_t, t + \delta t) = \begin{pmatrix} \left.\frac{\partial I(f(\bar{x}_1, \bar{\mu}), t+\delta t)}{\partial \bar{\mu}}\right|_{\bar{\mu}=\bar{\mu}_t} \\ \vdots \\ \left.\frac{\partial I(f(\bar{x}_N, \bar{\mu}), t+\delta t)}{\partial \bar{\mu}}\right|_{\bar{\mu}=\bar{\mu}_t} \end{pmatrix} \tag{4.5}$$

es la derivada parcial de los niveles de gris con respecto a los parámetros de movimiento.

Introduciendo (4.4) en (4.3), tenemos

$$\min_{\delta\bar{\mu}_a} \| \mathbf{M}(\bar{\mu}_t, t + \delta t)\delta\bar{\mu}_a + \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t + \delta t) - \mathbf{I}_r(\bar{x}) \|^2 . \tag{4.6}$$

Esta minimización se puede resolver reescribiéndola como sigue (obsérvese que es lineal en las incógnitas $\delta\bar{\mu}_a$):

$$\mathbf{M}(\bar{\mu}_t, t + \delta t)\delta\bar{\mu}_a = \mathcal{E}(t + \delta t), \tag{4.7}$$

donde $\mathcal{E}(t + \delta t) = \mathbf{I}_r(\bar{x}) - \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t + \delta t)$. El incremento de parámetros, $\delta\bar{\mu}_a$ se puede calcular como una solución mínimo cuadrática:

$$\delta\bar{\mu}_a = (\mathbf{M}(\bar{\mu}_t, t + \delta t)^\top \mathbf{M}(\bar{\mu}_t, t + \delta t))^{-1} \mathbf{M}(\bar{\mu}_t, t + \delta t)\mathcal{E}(t + \delta t), \tag{4.8}$$

$\mathbf{M}(\bar{\mu}_t, t + \delta t)$ no se puede calcular directamente (porque no conocemos la forma analítica de la función $I(\bar{x}, t)$) pero se puede hacer a partir de los gradientes de la imagen en el instante $t + \delta t$:

$$\mathbf{M}(\bar{\mu}_t, t + \delta t) = \begin{pmatrix} \left[\left.\frac{\partial I(\bar{y}, t+\delta t)}{\partial \bar{y}}\right|_{\bar{y}=f(\bar{x}_1, \bar{\mu}_t)}\right]^\top \left[\left.\frac{\partial f(\bar{x}_1, \bar{\mu})}{\partial \bar{\mu}}\right|_{\bar{\mu}=\bar{\mu}_t}\right] \\ \vdots \\ \left[\left.\frac{\partial I(\bar{y}, t+\delta t)}{\partial \bar{y}}\right|_{\bar{y}=f(\bar{x}_N, \bar{\mu}_t)}\right]^\top \left[\left.\frac{\partial f(\bar{x}_N, \bar{\mu})}{\partial \bar{\mu}}\right|_{\bar{\mu}=\bar{\mu}_t}\right] \end{pmatrix} . \tag{4.9}$$

El algoritmo de Lucas y Kanade es iterativo (ver algoritmo 4.1) y, comenzando con los parámetros de movimiento estimados en la imagen anterior de la secuencia, habrá que iterar hasta la convergencia sobre la imagen actual.

El orden de complejidad en número de operaciones de una iteración del algoritmo de Lucas y Kanade se muestra en el cuadro 4.1. Recordemos que el número de parámetros de movimiento es n y N es el número de píxeles de la región objetivo. La principal limitación de este algoritmo, cuando se usa en el seguimiento de objetos, es el coste computacional por iteración. Podemos decir que el orden de complejidad del algoritmo es $O(n^2N)$ ya que no puede ocurrir que el número de píxeles N sea menor que el número de parámetros de movimiento a estimar, n. Por tanto, la multiplicación $\mathbf{M}^\top\mathbf{M}$ (paso 6 del algoritmo) es la que se lleva el mayor número de operaciones y determina el coste de una iteración. La conclusión que podemos extraer es que al tener que recalcular $\mathbf{M}$ para cada imagen de entrada, tendremos que calcular también $\mathbf{M}^\top\mathbf{M}$ en cada imagen.

- **En cada iteración:**
  1. Rectificar $I(\bar{z}, t+\delta t)$ para calcular $\mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  2. Calcular $\mathcal{E}(t+\delta t) = \mathbf{I}_r(\bar{x}) - \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  3. Calcular el gradiente $\frac{\partial I(\bar{y}, t+\delta t)}{\partial \bar{y}}$ y rectificarlo empleando $f(\bar{x}, \bar{\mu}_t)$ para ajustarlo al tamaño de $\mathbf{I}_r(\bar{x})$.
  4. Evaluar el Jacobiano del modelo de movimiento $\left[\frac{\partial f(\bar{x}, \bar{\mu})}{\partial \bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t}\right]$ para todos los píxeles en $\mathcal{R}$ y calcular $\mathbf{M}(\bar{\mu}_t, t+\delta t)$.
  5. Calcular $\mathbf{M}(\bar{\mu}_t, t+\delta t)^\top \mathcal{E}(t+\delta t)$.
  6. Calcular $\mathbf{M}(\bar{\mu}_t, t+\delta t)^\top \mathbf{M}(\bar{\mu}_t, t+\delta t)$.
  7. Usando (4.8) calcular $\delta\bar{\mu}_a$.
  8. Actualizar $\bar{\mu}_{t+\delta t} = \bar{\mu}_t + \delta\bar{\mu}_a$.

Algoritmo 4.1: Algoritmo de Lucas y Kanade

Paso (1)	Paso (2)	Paso (3)	Paso (4)	Paso (5)	Paso (6)
$O(nN)$	$O(N)$	$O(nN)$	$O(nN)$	$O(nN)$	$O(n^2N)$

Paso (7)	Paso (8)	Total
$O(n^3)$	$O(n)$	$O(n^2N + n^3)$

Cuadro 4.1: Coste computacional en número de operaciones del algoritmo de Lucas y Kanade.

4.1.2. Algoritmo de factorización del Jacobiano de Hager y Belhumeur

La principal diferencia entre el algoritmo de Hager y Belhumeur [40] con el de Lucas y Kanade [67] es la eficiencia. El primero es mucho más eficiente debido a que la mayor parte del Jacobiano de la imagen con respecto a los parámetros de movimiento se precalcula. En esta sección explicaremos cómo se logra incrementar la eficiencia y a qué precio.

En el caso de Hager y Belhumeur la forma de realizar la aproximación de Taylor de primer orden al término $\mathbf{I}(f(\bar{x}, \bar{\mu}_t + \delta\bar{\mu}_a), t + \delta t)$ en la minimización de (4.3) es diferente a la de Lucas y Kanade. No sólo se realiza el desarrollo en los parámetros de movimiento $\bar{\mu}$, sino que también en la coordenada temporal t, alrededor del punto $(\bar{\mu}_t,t)$:

$$\mathbf{I}(f(\bar{x}, \bar{\mu}_t + \delta\bar{\mu}_a), t + \delta t) \approx \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t) + \mathbf{M}(\bar{\mu}_t, t)\delta\bar{\mu}_a + \mathbf{I}_t(f(\bar{x}, \bar{\mu}_t), t)\delta t, \quad (4.10)$$

donde

$$\mathbf{M}(\bar{\mu}_t, t) = \begin{pmatrix} \frac{\partial I(f(\bar{x}_1,\bar{\mu}),t)}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t} \\ \vdots \\ \frac{\partial I(f(\bar{x}_N,\bar{\mu}),t)}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t} \end{pmatrix} = \begin{pmatrix} \left[\frac{\partial I(\bar{y},t)}{\partial\bar{y}}\Big|_{\bar{y}=f(\bar{x}_1,\bar{\mu}_t)}\right]\left[\frac{\partial f(\bar{x}_1,\bar{\mu})}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t}\right] \\ \vdots \\ \left[\frac{\partial I(\bar{y},t)}{\partial\bar{y}}\Big|_{\bar{y}=f(\bar{x}_N,\bar{\mu}_t)}\right]\left[\frac{\partial f(\bar{x}_N,\bar{\mu})}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t}\right] \end{pmatrix}, \quad (4.11)$$

es la derivada parcial de los niveles de gris con respecto a los parámetros de movimiento. Y por otro lado,

$$\mathbf{I}_t(f(\bar{x}, \bar{\mu}_t), t) = \begin{pmatrix} \frac{\partial I(f(\bar{x}_1,\bar{\mu}),t)}{\partial t} \\ \vdots \\ \frac{\partial I(f(\bar{x}_N,\bar{\mu}),t)}{\partial t} \end{pmatrix} \quad (4.12)$$

es la derivada de los niveles de gris con respecto al tiempo, que se puede aproximar mediante

$$\mathbf{I}_t(f(\bar{x}, \bar{\mu}_t), t)\delta t \approx \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t + \delta t) - \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t). \quad (4.13)$$

Hager y Belhumeur propusieron un procedimiento eficiente para calcular los Jacobianos de las imágenes evitando el cálculo de los gradientes $\frac{\partial I(\bar{y},t+\delta t)}{\partial\bar{y}}\Big|_{\bar{y}=f(\bar{x},\bar{\mu}_t)}$ en cada imagen de la secuencia, tal y como ocurre con el algoritmo de Lucas y Kanade. Esto se logra expresando estos gradientes en función del gradiente de la plantilla de referencia.

Si asumimos que la estimación actual de los parámetros de movimiento es exacta, $\bar{\mu}_t = \bar{\mu}_t^*$, entonces derivando con respecto a $\bar{x}$ en ambos lados de la ecuación (4.1) obtenemos:

$$\left[\frac{\partial I_r(\bar{x})}{\partial\bar{x}}\right]^\top = \left[\frac{\partial I(\bar{y},t)}{\partial\bar{y}}\Big|_{\bar{y}=f(\bar{x},\bar{\mu}_t)}\right]^\top \left[\frac{\partial f(\bar{x},\bar{\mu}_t)}{\partial\bar{x}}\right] \quad (4.14)$$

Las derivadas parciales de los valores de gris con respecto a los parámetros de movimiento se pueden expresar ahora como:

$$\left[\frac{\partial I(f(\bar{x},\bar{\mu}),t)}{\partial \bar{\mu}}\right]_{\bar{\mu}=\bar{\mu}_t} = \left[\frac{\partial I_r(\bar{x})}{\partial \bar{x}}\right]^{\top} \left[\frac{\partial f(\bar{x},\bar{\mu}_t)}{\partial \bar{x}}\right]^{-1} \left[\frac{\partial f(\bar{x},\bar{\mu})}{\partial \bar{\mu}}\bigg|_{\bar{\mu}=\bar{\mu}_t}\right], \tag{4.15}$$

lo que significa que $\mathbf{M}(\bar{\mu},t)$, en cualquier instante de tiempo, sólo depende del modelo de movimiento y de las derivadas espaciales (gradientes) de la plantilla de referencia (que son constantes en toda la secuencia). Luego, $\mathbf{M}(\bar{\mu},t)$ será independiente de t y lo denotaremos por $\mathbf{M}(\bar{\mu})$. Desafortunadamente, en general, los otros dos términos en la ecuación (4.15), $\left[\frac{\partial f}{\partial \bar{x}}\right]^{-1}$ y $\left[\frac{\partial f}{\partial \bar{\mu}}\right]$, no son constantes. Pero, si se elige f de tal forma que se verifique

$$\left[\frac{\partial f(\bar{x},\bar{\mu})}{\partial \bar{x}}\right]^{-1} \left[\frac{\partial f(\bar{x},\bar{\mu})}{\partial \bar{\mu}}\right] = \mathbf{\Gamma}(\bar{x})\mathbf{\Sigma}(\bar{\mu}), \tag{4.16}$$

donde $\mathbf{\Gamma}(\bar{x})$ es una matriz que sólo depende de las coordenadas del píxel y $\mathbf{\Sigma}(\bar{\mu})$ es una matriz que depende sólo de los parámetros de movimiento, entonces podremos escribir $\mathbf{M}$ como

$$\mathbf{M}(\bar{\mu}) = \begin{pmatrix} \left[\frac{\partial I_r(\bar{x})}{\partial \bar{x}}\Big|_{\bar{x}=\bar{x}_1}\right]^{\top} \mathbf{\Gamma}(\bar{x}_1) \\ \vdots \\ \left[\frac{\partial I_r(\bar{x})}{\partial \bar{x}}\Big|_{\bar{x}=\bar{x}_N}\right]^{\top} \mathbf{\Gamma}(\bar{x}_N) \end{pmatrix} \mathbf{\Sigma}(\bar{\mu}) = \mathbf{M}_0\mathbf{\Sigma}(\bar{\mu}), \tag{4.17}$$

donde $\mathbf{M}_0$ es una matriz $N \times k$ constante y $\mathbf{\Sigma}$ es una matriz $k \times n$.

Ahora la solución mínimo cuadrática será de la forma

$$\delta\bar{\mu}_a = (\mathbf{\Sigma}(\bar{\mu}_t)^{\top}\mathbf{\Lambda}\mathbf{\Sigma}(\bar{\mu}_t))^{-1}\mathbf{\Sigma}(\bar{\mu}_t)^{\top}\mathbf{M}_0^{\top}\mathcal{E}(\bar{x},t+\delta t); \tag{4.18}$$

donde $\mathcal{E}(t+\delta t) = \mathbf{I}_r(\bar{x}) - \mathbf{I}(f(\bar{x},\bar{\mu}_t),t+\delta t)$ y $\mathbf{\Lambda} = \mathbf{M}_0^{\top}\mathbf{M}_0$ es una matriz constante, que se puede precalcular antes del seguimiento. El tamaño de $\mathbf{\Sigma}$ depende de la factorización (4.16), pero en general es pequeño y del orden del número de parámetros de movimiento, n.

$\mathbf{M}_0$ es nuestro conocimiento *a priori* sobre la estructura de la región objetivo, esto es, sobre cómo varía el valor de gris de cada píxel cuando el objeto se mueve. Representa la información aportada por cada píxel de la plantilla al proceso de seguimiento. Es importante tener en cuenta que cuando $\mathbf{M}_0^{\top}\mathbf{M}_0$ es singular no se pueden estimar los parámetros de movimiento, lo que se puede ver como una generalización del llamado *problema de apertura* en la estimación del flujo óptico.

El algoritmo resultante (ver algoritmo 4.2) permite realizar la alineación incremental de imágenes de forma iterativa. El orden de complejidad en número de operaciones del pre-cálculo de una iteración del algoritmo de Hager y Belhumeur se muestra en el cuadro 4.2 y el de una iteración del mismo en el cuadro 4.3.

- **Pre-cálculo:**
  1. Calcular y almacenar $\mathbf{M}_0$.
  2. Calcular y almacenar $\mathbf{\Lambda}$.
- **En cada iteración:**
  1. Rectificar $I(\bar{z}, t+\delta t)$ para calcular $\mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  2. Calcular $\mathcal{E}(t+\delta t) = \mathbf{I}_r(\bar{x}) - \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  3. Calcular $\mathbf{\Sigma}(\bar{\mu})$.
  4. Calcular $\mathbf{\Sigma}(\bar{\mu})^\top \mathbf{M}_0^\top$.
  5. Calcular $\mathbf{\Sigma}(\bar{\mu})^\top \mathbf{M}_0^\top \mathcal{E}(t+\delta t)$.
  6. Calcular $\mathbf{\Sigma}(\bar{\mu})^\top \mathbf{\Lambda} \mathbf{\Sigma}(\bar{\mu})$.
  7. Usando (4.18) calcular $\delta\bar{\mu}_a$.
  8. Actualizar $\bar{\mu}_{t+\delta t} = \bar{\mu}_t + \delta\bar{\mu}_a$.

Algoritmo 4.2: Algoritmo de Hager y Belhumeur, versión general.

Paso (1)	Paso (2)	Total
$O(kN)$	$O(k^2N)$	$O(k^2N)$

Cuadro 4.2: Coste computacional del pre-cálculo del algoritmo de Hager y Belhumeur cuando $\mathbf{\Sigma}(\bar{\mu})$ no tiene inversa. Se muestra el orden de complejidad en número de operaciones.

Paso (1)	Paso (2)	Paso (3)	Paso (4)	Paso (5)	Paso (6)
$O(nN)$	$O(N)$	$O(kn)$	$O(knN)$	$O(nN)$	$O(nk^2+n^2k)$

Paso (7)	Paso (8)	Total
$O(n^3)$	$O(n)$	$O(knN+n^3+nk^2+n^2k)$

Cuadro 4.3: Coste computacional de una iteración del algoritmo de Hager y Belhumeur cuando $\mathbf{\Sigma}(\bar{\mu})$ no tiene inversa. Se muestra el orden de complejidad en número de operaciones.

El caso más favorable en coste computacional se produce cuando $\mathbf{\Sigma}(\bar{\mu})$ tiene inversa, lo que depende del modelo de movimiento. Y en ese caso la ecuación (4.18) se simplifica y resulta

$$\delta\bar{\mu}_a = \mathbf{\Sigma}(\bar{\mu})^{-1}\mathbf{\Lambda}^{-1}\mathbf{M}_0^{\top}\mathcal{E}(\bar{x}, t+\delta t), \tag{4.19}$$

donde ahora toda la matriz $\mathbf{\Lambda}^{-1}\mathbf{M}_0^{\top}$ se puede precalcular. Por tanto, el algoritmo resultante cambia ligeramente y es más eficiente (ver algoritmo 4.3). El coste com-

- **Pre-cálculo:**
  1. Calcular y almacenar $\mathbf{M}_0$.
  2. Calcular y almacenar $\mathbf{\Lambda}$.
  3. Calcular y almacenar $\mathbf{\Lambda}^{-1}\mathbf{M}_0^{\top}$.
- **En cada iteración:**
  1. Rectificar $I(\bar{z}, t+\delta t)$ para calcular $\mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  2. Calcular $\mathcal{E}(t+\delta t) = \mathbf{I}_r(\bar{x}) - \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  3. Calcular $\mathbf{\Lambda}^{-1}\mathbf{M}_0^{\top}\mathcal{E}(t+\delta t)$.
  4. Calcular $\mathbf{\Sigma}(\bar{\mu})^{-1}$.
  5. Usando (4.19) calcular $\delta\bar{\mu}_a$.
  6. Actualizar $\bar{\mu}_{t+\delta t} = \bar{\mu}_t + \delta\bar{\mu}_a$.

Algoritmo 4.3: Algoritmo de Hager y Belhumeur, versión eficiente

putacional del algoritmo de factorización del Jacobiano cuando $\mathbf{\Sigma}(\bar{\mu})$ tiene inversa se muestra en los cuadros 4.4 y 4.5

Paso (1)	Paso (2)	Paso (3)	Total
$O(kN)$	$O(k^2N)$	$O(k^3+k^2N)$	$O(k^2N+k^3)$

Cuadro 4.4: Coste computacional del pre-cálculo del algoritmo de Hager y Belhumeur cuando $\mathbf{\Sigma}(\bar{\mu})$ tiene inversa. Se muestra el orden de complejidad en número de operaciones.

Paso (1)	Paso (2)	Paso (3)	Paso (4)	Paso (5)	Paso (6)	Total
$O(nN)$	$O(N)$	$O(kN)$	$O(k^3)$	$O(nk)$	$O(n)$	$O(nN+kN+k^3)$

Cuadro 4.5: Coste computacional de una iteración del algoritmo de Hager y Belhumeur cuando $\Sigma(\bar{\mu})$ tiene inversa. Se muestra el orden de complejidad en número de operaciones.

En el caso del algoritmo de factorización del Jacobiano presentado en esta sección, el coste computacional por iteración es mucho menor que el originalmente

propuesto por Lucas y Kanade. Sin embargo, la principal limitación proviene de que la factorización realizada en (4.16), en general, no siempre es posible.

4.1.3. Algoritmo composicional de Shum y Szelisky

El problema de seguimiento se puede afrontar de forma diferente. Shum y Szelisky propusieron en [81] una aproximación por composición en la que se busca una actualización, $\delta\bar{\mu}_c$, a los parámetros de movimiento de la forma siguiente:

$$I_r(\bar{x}) = I(f(\bar{x}, \bar{\mu}_{t+\delta t}), t+\delta t) = I(f(g(\bar{x}, \bar{\mu}_I + \delta\bar{\mu}_c), \bar{\mu}_t), t+\delta t), \tag{4.20}$$

donde $g(\bar{x}, \bar{\mu}_I) = \bar{x}$. En el algoritmo propuesto se elige g de tal forma que los parámetros de la transformación identidad para g sean $\bar{\mu}_I = 0$.

La minimización de la ecuación (4.2) se puede reescribir ahora como

$$\min_{\delta\bar{\mu}_c} \parallel \mathbf{I}(f(g(\bar{x}, \bar{\mu}_I + \delta\bar{\mu}_c), \bar{\mu}_t), t+\delta t) - \mathbf{I}_r(\bar{x}) \parallel^2, \tag{4.21}$$

Y empleando un desarrollo en serie de Taylor de primer orden alrededor de $\bar{\mu}_I$ obtenemos

$$\mathbf{I}(f(g(\bar{x}, \bar{\mu}_I + \delta\bar{\mu}_c), \bar{\mu}_t), t+\delta t) \approx \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t) + \mathbf{M}_{cs}(\bar{\mu}_t, t+\delta t)\delta\bar{\mu}_c \tag{4.22}$$

Introduciendo (4.22) en (4.21), resulta

$$\min_{\delta\bar{\mu}_c} \parallel \mathbf{M}_{cs}(\bar{\mu}_t, t+\delta t)\delta\bar{\mu}_c + \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t) - \mathbf{I}_r(\bar{x}) \parallel^2, \tag{4.23}$$

que es lineal en las incógnitas, $\delta\bar{\mu}_c$, y se puede resolver por mínimos cuadrados

$$\delta\bar{\mu}_c = (\mathbf{M}_{cs}(\bar{\mu}_t, t+\delta t)^\top \mathbf{M}_{cs}(\bar{\mu}_t, t+\delta t))^{-1} \mathbf{M}_{cs}(\bar{\mu}_t, t+\delta t)\mathcal{E}(t+\delta t), \tag{4.24}$$

donde $\mathcal{E}(t+\delta t) = \mathbf{I}_r(\bar{x}) - \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$. $\mathbf{M}_{cs}(\bar{\mu}_t, t+\delta t)$ se puede calcular empleando los gradientes espaciales de los píxeles en la región objetivo

$$\mathbf{M}_{cs}(\bar{\mu}_t, t+\delta t) = \begin{pmatrix} \left[\frac{\partial I(f(\bar{y},\bar{\mu}_t),t+\delta t)}{\partial \bar{y}}\Big|_{\bar{y}=g(\bar{x}_1,\bar{\mu}_I)}\right]^\top \left[\frac{\partial g(\bar{x}_1,\bar{\mu})}{\partial \bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_I}\right] \\ \vdots \\ \left[\frac{\partial I(f(\bar{y},\bar{\mu}_t),t+\delta t)}{\partial \bar{y}}\Big|_{\bar{y}=g(\bar{x}_N,\bar{\mu}_I)}\right]^\top \left[\frac{\partial g(\bar{x}_N,\bar{\mu})}{\partial \bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_I}\right] \end{pmatrix}, \tag{4.25}$$

donde $\frac{\partial I(f(\bar{y},\bar{\mu}_t),t+\delta t)}{\partial \bar{y}}$ es el gradiente de la imagen rectificada en el instante t con los parámetros de movimiento $\bar{\mu}_t$. Los pasos del algoritmo resultante se muestran en el algoritmo 4.4.

El coste computacional del pre-cálculo de una iteración del algoritmo de Shum y Szelisky (ver algoritmo 4.4) se muestra en el cuadro 4.6 y el de una iteración del mismo en el cuadro 4.7. Comparando en algoritmo de Shum y Szelisky [81] con el

- **Pre-cálculo:**
  1. Evaluar el Jacobiano $\left.\frac{\partial g(\bar{x},\bar{\mu})}{\partial \bar{\mu}}\right|_{\bar{\mu}=\bar{\mu}_I}$ para todos los píxeles en $\mathcal{R}$.
- **En cada iteración:**
  1. Rectificar $I(\bar{z}, t+\delta t)$ para calcular $\mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  2. Calcular $\mathcal{E}(t+\delta t) = \mathbf{I}_r(\bar{x}) - \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  3. Calcular el gradiente de la imagen rectificada para obtener $\left.\frac{\partial I(f(\bar{y},\bar{\mu}_t),t+\delta t)}{\partial \bar{y}}\right|_{\bar{y}=g(\bar{x}_N,\bar{\mu}_I)}$.
  4. Calcular $\mathbf{M}_{cs}(\bar{\mu}_t, t+\delta t)^\top \mathcal{E}(t+\delta t)$.
  5. Calcular $\mathbf{M}_{cs}(\bar{\mu}_t, t+\delta t)^\top \mathbf{M}_{cs}(\bar{\mu}_t, t+\delta t)$.
  6. Empleando (4.24) calcular $\delta\bar{\mu}_c$.
  7. Actualizar el modelo de movimiento, $f(\bar{x}, \bar{\mu}_{t+\delta t}) = f(g(\bar{x}, \delta\bar{\mu}_c), \bar{\mu}_t)$.

Algoritmo 4.4: Algoritmo de Shum y Szelisky.

de Lucas y Kanade [67], eliminamos el cálculo del Jacobiano del modelo de movimiento (que se puede precalcular) pero aún necesitamos calcular el gradiente de $I(f(\bar{x}, \bar{\mu}_t), t+\delta t)$ (paso 3. del algoritmo) en cada imagen de la secuencia y por tanto recalcular $\mathbf{M}_{cs}$. Si tenemos en cuenta que los pasos 5 y 6 involucran a $\mathbf{M}_{cs}$ y que marcan el coste computacional del algoritmo, se entiende por qué es tan importante poder precalcular $\mathbf{M}_{cs}$. Este algoritmo no es tan eficiente como la aproximación de Hager y Belhumeur [40]. Otra limitación es que el conjunto de modelos de movimiento que se pueden emplear deben verificar $f(\bar{x}, \bar{\mu}_{t+\delta t}) = f(g(\bar{x}, \bar{\mu}_I + \delta\bar{\mu}_c), \bar{\mu}_t)$, que no es una condición tan fácil de satisfacer como pueda parecer en un principio.

Paso (1)	Total
$O(nN)$	$O(nN)$

Cuadro 4.6: Coste computacional del pre-cálculo del algoritmo de Shum y Szelisky. Se muestra el orden de complejidad en número de operaciones.

Paso (1)	Paso (2)	Paso (3)	Paso (4)	Paso (5)	Paso (6)	Paso (7)	Total
$O(nN)$	$O(N)$	$O(N)$	$O(nN)$	$O(n^2N)$	$O(n^3)$	$O(n)$	$O(n^2N + n^3)$

Cuadro 4.7: Coste computacional de una iteración del algoritmo de Shum y Szelisky. Se muestra el orden de complejidad en número de operaciones.

4.1.4. Algoritmo composicional directo de Baker

Simon Baker propuso una aproximación composicional [4] diferente de la aproximación de Shum y Szelisky, presentada en la sección 4.1.3, en que g y f ahora son la misma función (ver figura 4.3).

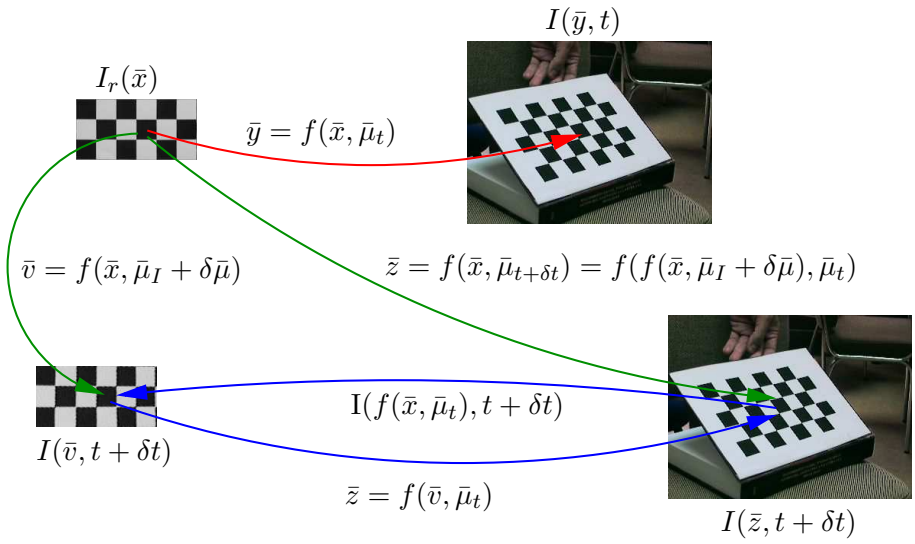


Figura 4.3: Elementos fundamentales en el algoritmo composicional directo de Baker. $I_r(\bar{x})$ es la imagen de la plantilla de referencia. $I(\bar{y},t)$ es la imagen capturada en el instante de tiempo t, $I(\bar{z},t+\delta t)$ es la imagen capturada en el instante de tiempo $t+\delta t$ e $I(\bar{v},t+\delta t)$ es la imagen rectificada en el instante $t+\delta t$ con los parámetros $\bar{\mu}_t$. A través de la función de movimiento f se pueden relacionar las coordenadas de una imagen con las de otra (conociendo los parámetros de movimiento) o se puede rectificar una imagen (sus niveles de gris) sobre las coordenadas de otra (*warping*). Se pretende encontrar los parámetros de movimiento para $f(\bar{x},\bar{\mu}_{t+\delta t})$, a partir de la composición de $\bar{v}=f(\bar{x},\bar{\mu}_I+\delta\bar{\mu})$ y $f(\bar{v},\bar{\mu}_t)$.

En este caso la minimización (4.2) se puede reescribir como

$$\min_{\delta\bar{\mu}_c} \parallel \mathbf{I}(f(f(\bar{x},\bar{\mu}_I+\delta\bar{\mu}_c),\bar{\mu}_t),t+\delta t)-\mathbf{I}_r(\bar{x}) \parallel^2, \tag{4.26}$$

donde $\bar{\mu}_I$ representa la identidad para la función f. Realizando un desarrollo en serie de Taylor de primer orden alrededor del punto $\bar{\mu}_I$ obtenemos

$$\mathbf{I}(f(f(\bar{x},\bar{\mu}_I+\delta\bar{\mu}_c),\bar{\mu}_t),t+\delta t)=\mathbf{I}(f(\bar{x},\bar{\mu}_t),t+\delta t)+\mathbf{M}_{cb}(\bar{\mu}_t,t+\delta t)\delta\bar{\mu}_c \tag{4.27}$$

Introduciendo (4.27) en (4.26), obtenemos la siguiente minimización

$$\min_{\delta\bar{\mu}_c} \parallel \mathbf{M}_{cb}(\bar{\mu}_t,t+\delta t)\delta\bar{\mu}_c+\mathbf{I}(f(\bar{x},\bar{\mu}_t),t+\delta t)-\mathbf{I}_r(\bar{x}) \parallel^2, \tag{4.28}$$

que es lineal en las incógnitas $\delta\bar{\mu}_c$. Se puede resolver por mínimos cuadrados

$$\bar{\mu}_c=(\mathbf{M}_{cb}(\bar{\mu}_t,t+\delta t)\mathbf{M}_{cb}(\bar{\mu}_t,t+\delta t)^\top)^{-1}\mathbf{M}_{cb}(\bar{\mu}_t,t+\delta t)\mathcal{E}(t+\delta t), \tag{4.29}$$

donde $\mathcal{E}(t+\delta t) = \mathbf{I}_r(\bar{x}) - \mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t)$ y $\mathbf{M}_{cb}(\bar{\mu}_t, t+\delta t)$ se puede calcular a partir de los gradientes de la imagen rectificada $I(f(\bar{y},\mu_t), t+\delta t)$,

$$\mathbf{M}_{cb}(\bar{\mu}_t, t+\delta t) = \begin{pmatrix} \left[\frac{\partial I(f(\bar{y},\bar{\mu}_t),t+\delta t)}{\partial \bar{y}}\Big|_{\bar{y}=g(\bar{x}_1,\bar{\mu}_I)}\right]^\top \left[\frac{\partial f(\bar{x}_1,\bar{\mu})}{\partial \bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_I}\right] \\ \vdots \\ \left[\frac{\partial I(f(\bar{y},\bar{\mu}_t),t+\delta t)}{\partial \bar{y}}\Big|_{\bar{y}=g(\bar{x}_N,\bar{\mu}_I)}\right]^\top \left[\frac{\partial f(\bar{x}_N,\bar{\mu})}{\partial \bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_I}\right] \end{pmatrix}. \tag{4.30}$$

Los pasos del algoritmo resultante se muestran en el algoritmo 4.5.

- **Pre-cálculo:**
  1. Evaluar el Jacobiano $\frac{\partial f(\bar{x},\bar{\mu})}{\partial \bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_I}$ para todos los píxeles en $\mathcal{R}$.
- **En cada iteración:**
  1. Rectificar $I(\bar{z}, t+\delta t)$ para calcular $\mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t)$.
  2. Calcular $\mathcal{E}(t+\delta t) = \mathbf{I}_r(\bar{x}) - \mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t)$.
  3. Calcular el gradiente de la imagen rectificada, para obtener $\frac{\partial I(f(\bar{x},\bar{\mu}_t),t+\delta t)}{\partial \bar{x}}$.
  4. Calcular $\mathbf{M}_{cs}(\bar{\mu}_t, t+\delta t)^\top \mathcal{E}(t+\delta t)$.
  5. Calcular $\mathbf{M}_{cb}(\bar{\mu}_t, t+\delta t)^\top \mathbf{M}_{cb}(\bar{\mu}_t, t+\delta t)$.
  6. Empleando (4.29) calcular $\delta\bar{\mu}_c$.
  7. Actualizar el modelo de movimiento, $f(\bar{x},\bar{\mu}_{t+\delta t}) = f(f(\bar{x},\delta\bar{\mu}_c),\bar{\mu}_t)$.

Algoritmo 4.5: Algoritmo composicional directo de Baker.

En este caso el coste computacional del algoritmo (ver algoritmo 4.5) es el mismo que el de Shum y Szelisky. Al igual que con el algoritmo de Shum y Szelisky, evitamos el cálculo del Jacobiano del modelo de movimiento pero todavía necesitamos calcular el gradiente de $I(f(\bar{x},\bar{\mu}_t), t+\delta t)$ para cada imagen de la secuencia.

Aunque no es evidente a partir de las descripciones de los algoritmos, existe una diferencia muy importante entre las aproximaciones de Baker y de Szelisky. Eligiendo la función g adecuadamente (de la forma explicada en la sección 4.1.3) el método de Shum y Szelisky se puede utilizar con cualquier modelo de movimiento basado en una homografía parametrizada (con la rotación y traslación en 3D por ejemplo). El algoritmo de Baker necesita: 1) que la función f sea cerrada bajo composición (paso 7) y 2) la existencia de un vector de parámetros de movimiento para la transformación identidad, $\bar{\mu}_I$, tal que $f(\bar{x},\bar{\mu}_I) = \bar{x}$. Esto es, el método de Baker necesita modelos de movimiento que formen semigrupos.

4.1.5. Algoritmo composicional inverso de Baker

Existe un segundo algoritmo composicional propuesto por Baker en [4]. Este es, a diferencia del composicional directo, tan eficiente como el método de Hager y Belhumeur. En este caso se desarrolla en serie de Taylor el término correspondiente a la plantilla de referencia en la ecuación (4.2). El lugar en el que se realiza el desarrollo nos permite llamar al algoritmo "inverso" en contraste con los otros dos algoritmos composicionales (ver figura 4.4).

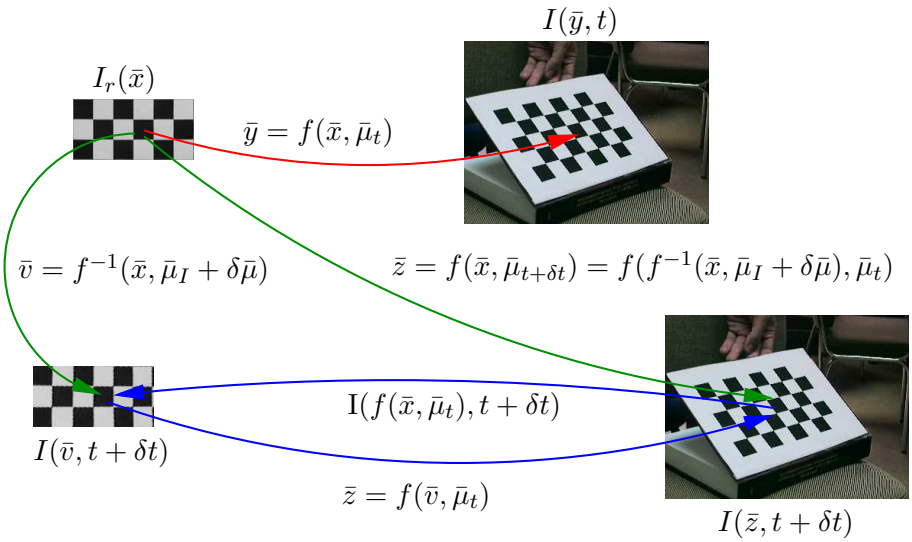


Figura 4.4: Elementos fundamentales en el algoritmo composicional inverso de Baker. $I_r(\bar{x})$ es la imagen de la plantilla de referencia. $I(\bar{y},t)$ es la imagen capturada en el instante de tiempo t, $I(\bar{z},t+\delta t)$ es la imagen capturada en el instante de tiempo $t+\delta t$. A través de la función de movimiento f se pueden relacionar las coordenadas de una imagen con las de otra (conociendo los parámetros de movimiento adecuados) o se puede rectificar una imagen (sus niveles de gris) sobre las coordenadas de otra (*warping*). Se pretende encontrar los parámetros de movimiento para $f(\bar{x},\bar{\mu}_{t+\delta t})$, a partir de la composición de $\bar{v}=f(\bar{x},\bar{\mu}_I+\delta\bar{\mu})$ y $f(\bar{v},\bar{\mu}_t)$.

En este algoritmo se pretende rectificar la imagen de la plantilla de referencia para alinearla, en el instante $t+\delta t$, con la imagen rectificada con los parámetros de movimiento anteriormente calculados, $\bar{\mu}_t$. Se calculará el incremento de parámetros de movimiento en el instante $t+\delta t$, que mediante composición con los anteriormente calculados en t, $\bar{\mu}_t$, permitan la alineación de la plantilla de referencia con la imagen rectificada. La minimización será por tanto

$$\delta\bar{\mu}_c = arg\min_{\delta\bar{\mu}} \parallel \mathbf{I}(f(f^{-1}(\bar{y},\bar{\mu}_I+\delta\bar{\mu}),\bar{\mu}_t),t+\delta t)-\mathbf{I}_r(\bar{y})) \parallel^2 . \tag{4.31}$$

La clave para la eficiencia del algoritmo consiste en un cambio de variable, $\bar{x}=f^{-1}(\bar{y},\bar{\mu}_I+\delta\bar{\mu}_c)$, en la ecuación (4.31) para intercambiar los papeles de la imagen rectificada y la plantilla de seguimiento, lo que nos permite reescribir la minimización

como

$$\min_{\delta\bar{\mu}_c} \| \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t + \delta t) - \mathbf{I}_r(f(\bar{x}, \bar{\mu}_I + \delta\bar{\mu}_c)) \|^2 . \tag{4.32}$$

Realizando un desarrollo en serie de Taylor de primer orden del término correspondiente a la plantilla de referencia alrededor de $\bar{\mu}_I$ obtenemos

$$\mathbf{I}_r(f(\bar{x}, \bar{\mu}_I + \delta\bar{\mu}_c)) \approx \mathbf{I}_r(f(\bar{x}, \bar{\mu}_I)) + \mathbf{M}_{cib}\delta\bar{\mu}_c. \tag{4.33}$$

La variación de los valores de gris de la plantilla con respecto a los parámetros de movimiento se representa mediante:

$$\mathbf{M}_{cib} = \begin{pmatrix} \frac{\partial I_r(f(\bar{x}_1,\bar{\mu}))}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_I} \\ \vdots \\ \frac{\partial I_r(f(\bar{x}_N,\bar{\mu}))}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_I} \end{pmatrix}. \tag{4.34}$$

Este Jacobiano sólo depende de los parámetros de movimiento en el instante inicial, $\bar{\mu}_I$, luego $\mathbf{M}_{cib}$ es una matriz constante. La minimización (4.32) se puede resolver para $\delta\bar{\mu}_c$ por mínimos cuadrados de la siguiente forma

$$\delta\bar{\mu}_c = (\mathbf{M}_{cib}^\top\mathbf{M}_{cib})^{-1}\mathbf{M}_{cib}\mathcal{E}(t + \delta t), \tag{4.35}$$

donde $\mathcal{E}(t + \delta t) = \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t + \delta t) - \mathbf{I}_r(\bar{x})$ y $\mathbf{M}_{cib}$ se puede calcular como sigue

$$\mathbf{M}_{cib} = \begin{pmatrix} \left[\frac{\partial I_r(\bar{z})}{\partial\bar{z}}\Big|_{\bar{z}=f(\bar{x}_1,\bar{\mu}_I)}\right]^\top \left[\frac{\partial f(\bar{x}_1,\bar{\mu})}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_I}\right] \\ \vdots \\ \left[\frac{\partial I_r(\bar{z})}{\partial\bar{z}}\Big|_{\bar{z}=f(\bar{x}_N,\bar{\mu}_I)}\right]^\top \left[\frac{\partial f(\bar{x}_N,\bar{\mu})}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_I}\right] \end{pmatrix}. \tag{4.36}$$

Los pasos del algoritmo resultante se pueden observar en el algoritmo 4.6. El coste computacional del pre-cálculo en el algoritmo composicional inverso se muestra en el cuadro 4.8 y el de una iteración del mismo en el cuadro 4.9.

Con este algoritmo el Jacobiano, $\mathbf{M}_{cib}$, se puede precalcular y con él la mayor parte de los cálculos en la ecuación (4.35) se realizan en una etapa previa al seguimiento (pasos 1, 2 y 3 del Pre-cálculo). Por otra parte, para poder poder aplicarlo necesitamos un modelo de movimiento, f, que sea cerrado bajo composición, invertible y que exista un vector de parámetros para la transformación identidad, $\bar{\mu}_I$. Por lo que buscamos modelos de movimiento que conformen un grupo.

4.1.6. Algoritmo de Jurie y Dhome

El algoritmo de Jurie y Dhome [62] también sigue la aproximación composicional inversa en la actualización de los parámetros de movimiento. Es tan eficiente como el composicional inverso de Baker y el aditivo de Hager y Belhumeur.

- **Pre-cálculo**:
  1. Calcular y almacenar $\mathbf{M}_{cib}$.
  2. Calcular y almacenar $(\mathbf{M}_{cib}^{\top}\mathbf{M}_{cib})^{-1}$.
  3. Calcular y almacenar $(\mathbf{M}_{cib}^{\top}\mathbf{M}_{cib})^{-1}\mathbf{M}_{cib}$.
- **En cada iteración**
  1. Rectificar $\mathbf{I}(\bar{z}, t+\delta t)$ para calcular $\mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  2. Calcular $\mathcal{E}(t+\delta t) = \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t) - \mathbf{I}_r(\bar{x})$.
  3. A partir de (4.35) calcular $\delta\bar{\mu}_c$.
  4. Actualizar el modelo de movimiento, $f(\bar{x}, \bar{\mu}_{t+\delta t}) = f(f^{-1}(\bar{x}, \bar{\mu}_I + \delta\bar{\mu}_c), \bar{\mu}_t)$.

Algoritmo 4.6: Algoritmo composicional inverso de Baker

Paso (1)	Paso (2)	Paso (3)	Total
$O(nN)$	$O(n^3 + n^2N)$	$O(nN)$	$O(n^3 + n^2N)$

Cuadro 4.8: Coste computacional del pre-cálculo del algoritmo composicional inverso de Baker. Se muestra el orden de complejidad en número de operaciones.

Paso (1)	Paso (2)	Paso (3)	Paso (4)	Total
$O(nN)$	$O(N)$	$O(nN)$	$O(n^2)$	$O(nN + n^2)$

Cuadro 4.9: Coste computacional de una iteración del algoritmo composicional inverso de Baker. Se muestra el orden de complejidad en número de operaciones.

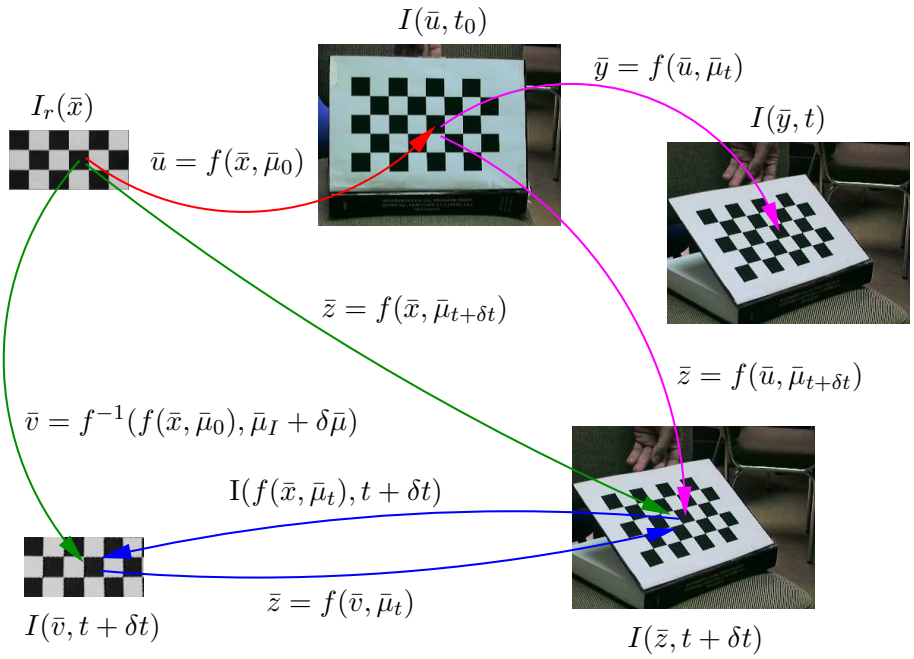


Figura 4.5: Elementos fundamentales en el algoritmo composicional de Jurie y Dhome. $I_r(\bar{x})$ es la imagen de la plantilla de referencia. $I(\bar{y}, t)$ es la imagen capturada en el instante de tiempo t, $I(\bar{z}, t + \delta t)$ es la imagen capturada en el instante de tiempo $t + \delta t$ y $I(\bar{u}, t_0)$ es la imagen capturada en el instante de tiempo t_0 y de la que se extrajo la plantilla de referencia. A través de la función de movimiento f se pueden relacionar las coordenadas de una imagen con las de otra (conociendo los parámetros de movimiento adecuados) o se puede rectificar una imagen (sus niveles de gris) sobre las coordenadas de otra (*warping*). Se pretende encontrar los parámetros de movimiento $\bar{\mu}_{t+\delta t}$, a partir de la composición de $\bar{v} = f^{-1}(f(\bar{x}, \bar{\mu}_0), \bar{\mu}_I + \delta\bar{\mu})$ y $f(\bar{v}, \bar{\mu}_t)$.

En este caso la ecuación de constancia del brillo se plantea sobre la imagen donde se tomó la plantilla de referencia, la imagen de referencia (ver figura 4.5), con lo que ahora tendremos

$$I_r(\bar{x}) = I(f(\bar{x}, \bar{\mu}_0), t_0) = I(f(\bar{x}, \bar{\mu}_t^*), t) \; \forall \bar{x} \in \mathcal{R}, \tag{4.37}$$

donde $\bar{\mu}_0$ son los parámetros de movimiento que nos llevan desde la plantilla de referencia a su posición en la imagen de referencia. La función de coste que se emplea para estimar los parámetros de movimiento en este caso es

$$\min_{\delta\bar{\mu}_c} \| \mathbf{I}(f(f^{-1}(f(\bar{x}, \bar{\mu}_0), \bar{\mu}_0 + \delta\bar{\mu}_c), \bar{\mu}_t), t + \delta t) - \mathbf{I}_r(\bar{x}) \|^2 . \tag{4.38}$$

Si ahora hacemos el cambio de variable $\bar{y} = f^{-1}(f(\bar{x}, \bar{\mu}_0), \bar{\mu}_0 + \delta\bar{\mu}_c)$ podemos reescribir la minimización (4.38) como

$$\min_{\delta\bar{\mu}_c} \| \mathbf{I}(f(\bar{y}, \bar{\mu}_t), t + \delta t) - \mathbf{I}_r(f^{-1}(f(\bar{y}, \bar{\mu}_0 + \delta\bar{\mu}_c), \bar{\mu}_0)) \|^2 . \tag{4.39}$$

Realizando un desarrollo en serie de Taylor de primer orden de la plantilla de referencia alrededor del punto $(\bar{\mu}_0)$ obtenemos

$$I_r(f^{-1}(f(\bar{y}, \bar{\mu}_0 + \delta\bar{\mu}_c), \bar{\mu}_0)) \approx I_r(\bar{y}) + \mathbf{M}_{cjd}\delta\bar{\mu}_c. \tag{4.40}$$

La variación de los valores de gris de la plantilla con respecto a los parámetros de movimiento se puede calcular mediante

$$\mathbf{M}_{cjd} = \begin{pmatrix} \left[\frac{\partial I_r(f^{-1}(f(\bar{y}_1, \bar{\mu}), \bar{\mu}_0))}{\partial \bar{\mu}} \Big|_{\bar{\mu}=\bar{\mu}_0} \right] \\ \vdots \\ \left[\frac{\partial I_r(f^{-1}(f(\bar{y}_N, \bar{\mu}), \bar{\mu}_0))}{\partial \bar{\mu}} \Big|_{\bar{\mu}=\bar{\mu}_0} \right] \end{pmatrix}. \tag{4.41}$$

Este Jacobiano sólo depende de los parámetros de movimiento en el instante inicial, $\bar{\mu}_0$, luego $\mathbf{M}_{cjd}$ es una matriz constante. La minimización (4.39) se puede resolver para $\delta\bar{\mu}_c$ por mínimos cuadrados de la siguiente forma

$$\delta\bar{\mu}_c = (\mathbf{M}_{cjd}^\top \mathbf{M}_{cjd})^{-1} \mathbf{M}_{cjd} \mathcal{E}(t + \delta t) = \mathbf{A}_h \mathcal{E}(t + \delta t), \tag{4.42}$$

donde $\mathcal{E}(t + \delta t) = \mathbf{I}(f(\bar{y}, \bar{\mu}_t), t + \delta t) - \mathbf{I}_r(\bar{y})$.

Jurie y Dhome en [62] no calculan explícitamente $\mathbf{M}_{cjd}$ sino que se calcula directamente $\mathbf{A}_h$ de forma numérica. Supongamos que la posición actual de la región de interés viene caracterizada por $\bar{\mu}_0$. Si perturbamos este vector de tal forma que $\bar{\mu}' = \bar{\mu}_0 + \delta\bar{\mu}_c$, la región de interés se mueve y se produce una variación de los niveles de gris tal que

$$\mathcal{E} = \mathbf{I}(f(\bar{x}, \bar{\mu}_0 + \delta\bar{\mu}_c), t_0) - \mathbf{I}(f(\bar{x}, \bar{\mu}_0), t_0) = \mathbf{I}(f(\bar{x}, \bar{\mu}_0 + \delta\bar{\mu}_c), t_0) - \mathbf{I}_r(\bar{x}). \tag{4.43}$$

Si calculamos N_p perturbaciones $\delta\bar{\mu}_c$, tendremos N_p parejas $(\mathcal{E}^k, \delta\bar{\mu}_c^k)$. Podremos obtener $\mathbf{A}_h$ de la siguiente forma

$$\mathbf{A}_h = \mathbf{Y}(\mathbf{H}^\top\mathbf{H})\mathbf{H}^\top, \tag{4.44}$$

donde $\mathbf{H} = (\mathcal{E}^1, \cdots, \mathcal{E}^{N_p})$ es una matriz $N_p \times N$, y $\mathbf{Y} = (\delta\bar{\mu}_c^1, \cdots, \delta\bar{\mu}_c^{N_p})$ es una matriz $n \times N_p$.

Por tanto el algoritmo de Jurie y Dhome [62] consiste en los pasos descritos en el algoritmo 4.7.

- **Pre-cálculo**:
  1. Calcular numéricamente y almacenar $\mathbf{A}_h$.
- **En cada iteración**
  1. Rectificar $\mathbf{I}(\bar{z}, t+\delta t)$ para calcular $\mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  2. Calcular $\mathcal{E}(t+\delta t) = \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t) - \mathbf{I}_r(\bar{x})$.
  3. A partir de (4.42) calcular $\delta\bar{\mu}_c$.
  4. Actualizar el modelo de movimiento, $f(\bar{x}, \bar{\mu}_{t+\delta t}) = f(f^{-1}(f(\bar{x}, \bar{\mu}_0), \bar{\mu}_0 + \delta\bar{\mu}_c), \bar{\mu}_t)$.

Algoritmo 4.7: Algoritmo de Jurie y Dhome

El coste computacional del pre-cálculo en el algoritmo se muestra en el cuadro 4.10 y el de una iteración del mismo en el cuadro 4.11. La diferencia fundamental de este algoritmo con el composicional inverso de Baker es el cálculo numérico del Jacobiano. En [62] Jurie y Dhome aseguran que su algoritmo converge desde posiciones más alejadas a la verdadera que el algoritmo de Hager y Belhumeur. Lo que podría explicarse por el hecho de que calcular $\mathbf{A}_h$ de forma numérica, con suficientes de deformaciones $\delta\bar{\mu}_c$, puede mejorar la aproximación de primer orden en la que descansa el algoritmo de Hager y Belhumeur.

Paso (1)	Total
$O(N_p n N)$	$(O(N_p n N)$

Cuadro 4.10: Coste computacional del pre-cálculo del algoritmo composicional de Jurie y Dhome. Se muestra el orden de complejidad en número de operaciones.

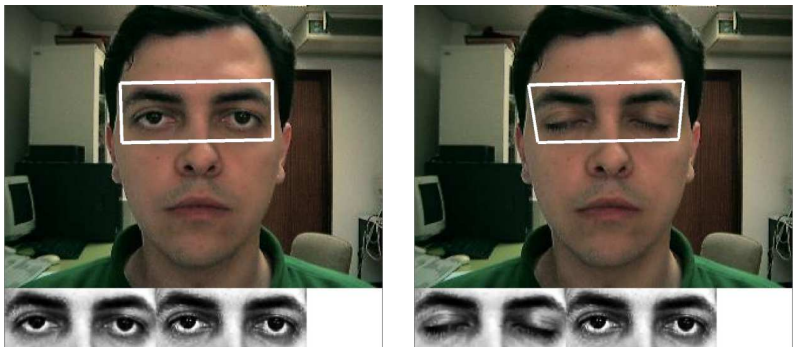
Paso (1)	Paso (2)	Paso (3)	Paso (4)	Total
$O(nN)$	$O(N)$	$O(nN)$	$O(n^2)$	$O(n^2 + nN)$

Cuadro 4.11: Coste computacional de una iteración del algoritmo de Jurie y Dhome. Se muestra el orden de complejidad en número de operaciones.

4.2. Alineación con movimiento no rígido

Uno de los mayores retos que afrontan los algoritmos de seguimiento visual es el ser robustos frente a los cambios en la apariencia del objetivo durante el seguimiento. Los cambios de apariencia pueden deberse a una variación de la iluminación, una oclusión o a un cambio en la posición o en la configuración del objetivo (por ejemplo, si el objeto es flexible).

Todos los algoritmos de la sección 4.1 se pueden emplear en el seguimiento rígido de planos e incluso de cualquier objeto que permanezca casi paralelo al plano imagen. En todos los casos anteriores tenemos una plantilla de referencia, una imagen previa del objeto a seguir, que nos permite estimar el movimiento. Imaginemos que estamos siguiendo un ojo empleando una plantilla con un ojo abierto. Parece claro que en el momento que el ojo se cierre nuestra estimación de los parámetros de movimiento se verá perturbada y, en última instancia, podríamos perder el objeto seguido. Continuando con el ejemplo, en la figura 4.6 (a) se estima correctamente el movimiento de los ojos abiertos con una plantilla tomada con la misma expresión, pero en (b), al no coincidir los ojos cerrados con la plantilla, el movimiento se estima incorrectamente.



(a) (b)

Figura 4.6: Seguimiento rígido de un movimiento no rígido. En esta figura se muestra el resultado del seguimiento con un modelo de movimiento proyectivo. En ambas imágenes se muestra la región objetivo recuadrada en blanco. Debajo de cada imagen, a la izquierda se encuentra la imagen rectificada y a la derecha la plantilla de referencia.

Los algoritmos de seguimiento tratan de adaptarse a estas variaciones modelando la apariencia del objetivo de diversas formas. Algunos emplean textura [58], estadísticas sobre el color [19] o la forma [55], o sobre ambas [8], otros utilizan modelos 3D con textura [64], y finalmente, la mayoría emplean modelos basados en subespacios lineales de textura [10, 40] o de textura y forma [21]. En esta sección repasaremos algunas aproximaciones a la alineación de imágenes con movimiento no rígido que emplean directamente los niveles de gris.

4.2.1. Modelos de movimiento locales

Black y Yacoob [9], plantearon una solución a la estimación del movimiento no rígido de las zonas de interés en el rostro, mediante el empleo de flujo óptico parametrizado y modelos de movimiento simples. En su trabajo muestran cómo el movimiento rígido de la cara se puede emplear para rectificar la imagen actual y conseguir una vista estabilizada del rostro. Esta vista estabilizada permite estimar los desplazamientos en la imagen de las partes interesantes de la cara (los ojos, la boca y las cejas) de forma relativa al movimiento de la cabeza en su conjunto. Aislando los movimientos de estas zonas del desplazamiento global de la cabeza se pueden reconocer expresiones faciales.

El rostro sin las partes más expresivas (boca, ojos y cejas) se considera como plano. Si F es el conjunto de todos los píxeles en la zona de movimiento rígido de la cara cuyas posiciones vienen dadas por $\bar{x}_i = (x_i, y_i)$, para todos ellos se elige un modelo de movimiento plano de 8 parámetros para su seguimiento y localización:

$$f(\bar{x}, \bar{\mu}) = \begin{pmatrix} a_1 & a_2 \\ a_4 & a_5 \end{pmatrix} \bar{x} + \begin{pmatrix} a_0 \\ a_3 \end{pmatrix} + \bar{x}\bar{p}^\top \bar{x} \quad \forall \bar{x} \in F, \tag{4.45}$$

donde $\bar{p} = (p_0 \quad p_1)^\top$ y el vector de parámetros de movimiento en este caso es $\bar{\mu} = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, p_0, p_1)$.

La estimación de los parámetros de movimiento del rostro se hace mediante la minimización robusta de la diferencia entre los niveles de gris de la cara en en el instante t y el instante $t + \delta t$:

$$\min_{\delta\bar{\mu}} \rho(\mathbf{I}(f(\bar{x}, \bar{\mu}_t + \delta\bar{\mu}), t + \delta t) - \mathbf{I}(\bar{x}, t), \sigma), \tag{4.46}$$

donde $\rho(r, \sigma) = \frac{r^2}{\sigma^2 + r^2}$ es un estadístico robusto que reduce la influencia de los valores atípicos en la solución. La minimización se realiza en varios niveles de resolución y en varias iteraciones por nivel.

Una vez se conocen los parámetros del movimiento rígido se estiman los parámetros de los modelos de deformación de los ojos, cejas y boca. Para cada ojo se emplea una región rectangular y simple traslación, sin embargo para las cejas y la boca se utiliza un modelo de movimiento afín modificado:

$$f(\bar{x}, \bar{\mu}) = \begin{pmatrix} a_1 & a_2 \\ a_4 & a_5 \end{pmatrix} \bar{x} + \begin{pmatrix} a_0 \\ a_3 \end{pmatrix} + \bar{x}^\top \begin{pmatrix} c & 0 \\ 0 & 0 \end{pmatrix} \bar{x}, \tag{4.47}$$

donde c es el coeficiente de curvatura de las cejas o la boca. El procedimiento de minimización será el mismo que para el movimiento rígido pero empleando las imágenes estabilizadas del rostro.

Una vez estimados los parámetros de movimiento de ojos, cejas y boca, se establecía un sistema de reglas sencillo que permitía clasificar las expresiones faciales en las prototípicas: enfado, disgusto, felicidad, sorpresa, tristeza y miedo. Aunque en su implementación procesaban una imagen cada 2 minutos, la idea original de separar el movimiento rígido del no rígido empleando alineación incremental de imágenes en todo el proceso, es muy interesante por su simplicidad.

4.2.2. Seguimiento de la apariencia: *Eigentracking*

Los modelos de subespacios lineales constituyen, posiblemente, la forma más extendida de representar de la apariencia (ver figura 4.7). Las imágenes del objetivo se encuentran sobre una variedad (*manifold*) de pocas dimensiones que representan los grados de libertad subyacentes en el objeto que aparece en las imágenes. Por ejemplo, las imágenes de un ojo se encuentran en un subespacio de tres dimensiones, una dimensión asociada con el grado de apertura del párpado y las otras con la orientación de la pupila (aunque también podemos considerarlo de dos dimensiones dado que el párpado siempre baja acompañando a la pupila). La popularidad de estos modelos radica en su simplicidad, su eficiencia computacional y en que se han estudiado en profundidad en el contexto de la estadística y el reconocimiento de patrones. En Visión por Computador se han empleado con éxito para reconocer objetos 3D con cambios en la orientación y posición [73], en la representación y reconocimiento de caras [6, 94], en el seguimiento con cambios de iluminación [40] o con movimientos no rígidos [10], y en el seguimiento de objetos deformables en 3D [91], entre otros.

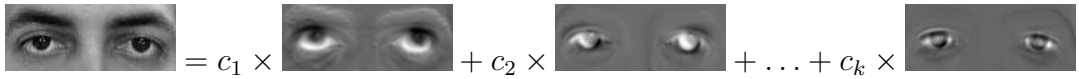


Figura 4.7: Análisis de Componentes Principales de imágenes. Permite la representación de la apariencia de un objeto como una combinación lineal de otras "imágenes"(en realidad vectores en un espacio N-dimensional) que representan los principales modos de variación de la apariencia del mismo.

Normalmente, la relación entre la imagen de entrada y la variedad no es lineal, aunque existen resultados muy útiles basados en relaciones lineales entre ambas. El Análisis de Componentes Principales (*PCA*) y el Análisis Factorial (*FA*) constituyen dos ejemplos de procedimientos lineales. En el Análisis de Componentes Principales la base del subespacio viene dada por los autovectores de la matriz de covarianzas muestral asociados con los mayores autovalores, que ha mostrado ser una excelente herramienta para la reducción de dimensiones en datos multivariantes. Por tanto, si una imagen se considera como un dato multivariante, el PCA puede ser una herramienta útil para la construcción de la variedad.

Se han propuesto múltiples extensiones a los modelos lineales convencionales a lo largo de los últimos años. Por ejemplo, el Análisis de Componentes Independientes (*ICA*) es un intento de conseguir la independencia entre los componentes de un vector multivariante [20]. En casos en los que los modelos de subespacios lineales no son suficientes, se pueden emplear las mezclas de modelos lineales [90, 33, 36] o técnicas de estimación de variedades no lineales como LLE (Locally Linear Embedding) [78].

Una de las mayores limitaciones del PCA es que necesita muestras normalizadas en el conjunto de imágenes de entrenamiento. Esto significa que las imágenes se

tienen que normalizar y alinear geométricamente tanto en la construcción del subespacio como en la proyección de las imágenes de entrada sobre él. Este problema se ha solucionado empleando subespacios [63] o procedimientos de proyección [80] que sean invariantes a estas transformaciones geométricas o también mediante la alineación robusta de imágenes [26, 84].

En el contexto de la alineación incremental de imágenes en presencia de deformaciones en el objeto seguido, Black y Jepson [10] plantearon un método basado en la separación entre apariencia, modelada mediante PCA, y movimiento rígido. En lugar de representar el objeto desde cada posible orientación, se representa el objeto desde unas pocas orientaciones. La idea es tratar de reconocer el objeto en otras orientaciones mediante una función de transformación parametrizada (rectificación) de la imagen de entrada y su proyección sobre el subespacio de apariencia (en este caso proveniente del PCA). Podemos resumir su método como una combinación de un modelo de movimiento más un subespacio lineal de la apariencia.

El algoritmo de Black y Jepson [10] es un procedimiento iterativo basado en descenso del gradiente y una métrica robusta con incrementos en el nivel de resolución. Sin embargo, en esta sección, vamos a presentar el desarrollo del *Eigentracking* con una función de coste basada en la distancia euclídea al cuadrado y con un único nivel de resolución para poder comparar los órdenes de complejidad de este algoritmo con los precedentes y subsiguientes.

Sea P la imagen del objeto en seguimiento. La *ecuación de constancia de los niveles de gris en el subespacio* se verifica para todos los píxeles en la región objetivo [10]:

$$I(f(\bar{x}, \bar{\mu}_t), t) = [\mathbf{B}\bar{c}(t)](\bar{x}) \quad \forall x \in P, \tag{4.48}$$

donde $\bar{x}$ es el vector de coordenadas de un punto en la imagen I, $\mathbf{B}$ es la matriz con los vectores de la base del subespacio, $\bar{c}$ es el vector de coeficientes del subespacio, e $I(f(\bar{x}, \bar{\mu}_t), t)$ es la imagen capturada en el instante t rectificada con el modelo de movimiento $f(\bar{x}, \bar{\mu})$ y parámetros $\bar{\mu}_t$. Por $[\mathbf{B}\bar{c}(t)](x)$ se denota el valor de $\mathbf{B}\bar{c}(t)$ para el píxel de posición $\bar{x}$ en la imagen. La matriz $\mathbf{B}$ es de dimensión $N \times l$, donde N es el número de píxeles en la región objetivo y l es el número de vectores en la base del subespacio. Intuitivamente (4.48) establece que la imagen rectificada suponiendo movimiento rígido, $I(f(\bar{x}, \bar{\mu}_t), t)$, se puede expresar como una combinación lineal de los vectores de la base del subespacio de apariencia, $\mathbf{B}$[1].

El seguimiento consiste en la estimación para cada imagen en la secuencia, de los parámetros de movimiento, $\bar{\mu}$, y de apariencia, $\bar{c}$, que minimizan la función de coste

$$E(\bar{\mu}, \bar{c}) = ||\mathbf{I}(f(\bar{x}, \bar{\mu}_t + \delta\bar{\mu}), t + \delta t) - \mathbf{B}\bar{c}(t + \delta t)||^2. \tag{4.49}$$

En general, minimizar (4.49) puede ser una tarea muy difícil ya que define una función de coste no convexa. Una vez más podemos linealizar la función de coste mediante un desarrollo en serie de Taylor de primer orden del término $I(f(\bar{x}, \bar{\mu}_t +$

[1]Asumimos, sin pérdida de generalidad, que la imagen media se ha incluido en la primera columna de $\mathbf{B}$.

$\delta\bar{\mu}), t+\delta t)$ alrededor de $\bar{\mu}_t$, obteniendo

$$\min_{\bar{\mu}} E(\bar{\mu},\bar{c}) = \min_{\bar{\mu}} ||\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)\delta\bar{\mu} + \mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t) - \mathbf{B}\bar{c}(t+\delta t)||^2, \quad (4.50)$$

donde $\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)$ es el Jacobiano que representa la variación de los niveles de gris con respecto a los parámetros de movimiento, o lo que es lo mismo

$$\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t) = \begin{pmatrix} \frac{\partial I(f(\bar{x}_1,\bar{\mu}),t+\delta t)}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t} \\ \vdots \\ \frac{\partial I(f(\bar{x}_N,\bar{\mu}),t+\delta t)}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t} \end{pmatrix} = \begin{pmatrix} \frac{\partial I(\bar{z},t+\delta t)}{\partial\bar{z}}\Big|_{\bar{z}=f(\bar{x}_1,\bar{\mu}_t)} & \frac{\partial f(\bar{x}_1,\bar{\mu})}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t} \\ \vdots & \\ \frac{\partial I(\bar{z},t+\delta t)}{\partial\bar{z}}\Big|_{\bar{z}=f(\bar{x}_N,\bar{\mu}_t)} & \frac{\partial f(\bar{x}_N,\bar{\mu})}{\partial\bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t} \end{pmatrix}. \quad (4.51)$$

Para minimizar (4.49), asumiremos $\bar{c}$ constante y encontraremos el mínimo para $E(\bar{\mu},\bar{c})$ con respecto a $\bar{\mu}$ (ver sección A.1 en el apéndice A):

$$\delta\bar{\mu} = (\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top \mathbf{M}_{et}(\bar{\mu}_t, t+\delta t))^{-1}\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top \mathcal{E}(t+\delta t), \quad (4.52)$$

donde $\mathcal{E} = [\mathbf{B}\bar{c}(t+\delta t) - \mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t)]$. A continuación encontraremos el mínimo de $E(\bar{\mu},\bar{c})$ con respecto $\bar{c}$ asumiendo $\bar{\mu}$ constante

$$\bar{c}(t+\delta t) = \mathbf{B}^\top[\mathbf{M}_{et}(\bar{\mu}, t+\delta t)\delta\bar{\mu} + \mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t)]. \quad (4.53)$$

Dado que el cálculo de $\bar{\mu}$ y de $\bar{c}$ dependen el uno del otro vamos a asumir que la apariencia del objeto en seguimiento varía lentamente. Esto nos permite emplear $\bar{c}(t)$ como punto de partida para el cálculo de $\bar{\mu}$ en el instante $t+\delta t$. Con lo que el resultado queda como se muestra en el algoritmo 4.8.

La minimización de (4.49) desarrollada hasta ahora, contempla los mismos pasos que el algoritmo de Lucas y Kanade, con el añadido del cálculo de los parámetros de apariencia y de la imagen reconstruida. El orden de complejidad del algoritmo resultante se muestran en el cuadro 4.12.

Black y Jepson [10] primero minimizan $E(\bar{\mu},\bar{c})$ con respecto a $\bar{c}$ dejando constante $\bar{\mu}$ y a continuación con el valor obtenido minimizan con respecto a $\bar{\mu}$ dejando $\bar{c}$ constante. Este proceso se realiza en todos los niveles de resolución de la pirámide comenzando con el nivel de menor resolución para cada imagen de la secuencia. En el artículo original [10] no se utiliza la distancia euclídea al cuadrado como hemos planteado en (4.49) sino una métrica robusta. Computacionalmente su algoritmo es bastante exigente ya que, por ejemplo, el Jacobiano de cada imagen de entrada tiene que calcularse una vez para cada nivel de la pirámide multi-resolución. El resultado es un algoritmo que no es eficiente en el número de operaciones pero que permite seguir objetos que cambian de apariencia.

4.2.3. Modelos Activos de Apariencia

En los Modelos Activos de Apariencia (AAM)[21] se emplea un modelo estadístico tanto de la forma como de los niveles de gris (de la textura). Se pueden emplear

- **En cada iteración:**

  1. Rectificar $I(\bar{z}, t+\delta t)$ para calcular $\mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  2. Calcular la imagen reconstruida $\mathbf{B}\bar{c}(t)$.
  3. Calcular $\mathcal{E}(t+\delta t) = \mathbf{B}\bar{c}(t) - \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)$.
  4. Calcular el gradiente $\frac{\partial I(\bar{y}, t+\delta t)}{\partial \bar{y}}$ y rectificarlo empleando $f(\bar{x}, \bar{\mu}_t)$ para ajustarlo al tamaño de $[\mathbf{B}\bar{c}(t)](x)$.
  5. Evaluar el Jacobiano del modelo de movimiento $\left[\frac{\partial f(\bar{x}, \bar{\mu})}{\partial \bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t}\right]$ para todos los píxeles en $\mathcal{R}$ y calcular $\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)$.
  6. Calcular $\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top \mathcal{E}(t+\delta t)$.
  7. Calcular $\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top \mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)$.
  8. Empleando (4.52) calcular $\delta\bar{\mu}_{et}$.
  9. Empleando (4.53) calcular $\bar{c}(t+\delta t)$.
  10. Actualizar $\bar{\mu}_{t+\delta t} = \bar{\mu}_t + \delta\bar{\mu}_{et}$.

Algoritmo 4.8: *Eigentracking* sin métrica robusta.

Paso (1)	Paso (2)	Paso (3)	Paso (4)	Paso (5)	Paso (6)
$\mathrm{O}(nN)$	$\mathrm{O}(lN)$	$\mathrm{O}(N)$	$\mathrm{O}(nN)$	$\mathrm{O}(nN)$	$\mathrm{O}(nN)$

Paso (7)	Paso (8)	Paso (9)	Paso (10)	Total
$\mathrm{O}(n^2N)$	$\mathrm{O}(n^3)$	$\mathrm{O}(lN+nN)$	$\mathrm{O}(n)$	$O(n^2N+n^3+lN)$

Cuadro 4.12: Coste computacional de una iteración del *Eigentracking*

para el seguimiento del movimiento no rígido de objetos. La ventaja del modelo combinado reside en que se obtienen modelos muy compactos (con pocos parámetros) y la clave para lograrlo reside en que se elimina la variabilidad de la forma de los objetos antes de modelar la variabilidad en la textura.

Los AAM parten de un conjunto de M imágenes de ejemplo en el que se han marcado N puntos de interés. Las coordenadas de los puntos de interés constituyen la forma del objeto y las M formas ejemplo se alinean con la forma media común a todas, $\bar{S}$. Las imágenes de ejemplo, por su parte, se rectifican mediante un modelo afín a trozos (una transformación afín para cada triángulo obtenido mediante el algoritmo de Delaunay sobre los puntos de interés) y se ponen en correspondencia sobre la forma media $\bar{S}$. El resultado es que tenemos las texturas de los M ejemplos del objeto perfectamente alineadas para construir el modelo de los niveles de gris mediante PCA.

Sea $\bar{s} = (\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{y}_1, \ldots, \mathbf{y}_N)^\top$ el vector con la forma, $\bar{g}$ el vector con los niveles de gris, $\bar{S}$ la forma media, y $\bar{G}$ los niveles de gris medios de los M ejemplos respectivamente. Entonces la forma y la textura se pueden representar mediante el siguiente modelo:

$$\bar{s} = \bar{S} + \Phi_s \bar{b}_s \tag{4.54}$$

$$\bar{g} = \bar{G} + \Phi_g \bar{b}_g, \tag{4.55}$$

donde Φ_s y Φ_g son las bases para la forma y los niveles de gris estimadas a partir de los ejemplos. Para obtener una parametrización combinada de forma y textura, $\bar{c}$, los valores de $\bar{b}_s$ y $\bar{b}_g$ se pueden fusionar mediante

$$\bar{b} = \begin{pmatrix} \mathbf{W}_s \bar{b}_s \\ \bar{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s \Phi_s^\top (\bar{s} - \bar{S}) \\ \Phi_g^\top (\bar{g} - \bar{G}) \end{pmatrix}, \tag{4.56}$$

donde $\mathbf{W}_s$ es una matriz de pesos necesaria para compensar la diferencia de escala entre las coordenadas y los niveles de gris. Realizando un nuevo PCA sobre los vectores $\bar{b}$ llegamos al modelo combinado

$$\bar{b} = \Phi_c \bar{c}. \tag{4.57}$$

Dado que los parámetros de forma y textura se encuentran centrados en la media el resultado combinado también lo estará. El nuevo modelo que permite representar forma y textura en función de un único vector de parámetros $\bar{c}$ tendrá la forma

$$\bar{s} = \bar{S} + \Phi_s \mathbf{W}_s^{-1} \Phi_{c,s} \bar{c}, \quad \bar{g} = \bar{G} + \Phi_g \Phi_{c,g} \bar{c}, \quad \Phi_c = \begin{pmatrix} \Phi_{c,t} \\ \Phi_{c,g} \end{pmatrix}. \tag{4.58}$$

Una imagen del objeto del que se dispone un AAM, representada por el vector de parámetros $\bar{c}_i$, se puede sintetizar generando la imagen sobre la forma media a partir de los niveles de gris en el vector $\bar{g}_i$ y a continuación rectificándola empleando las coordenadas dadas por $\bar{s}_i$.

Para generar las coordenadas de los puntos en una imagen con los AAM se emplea un modelo de movimiento

$$\bar{z} = f(\bar{y}, \bar{\mu}), \tag{4.59}$$

donde $\bar{y}$ son los puntos en el sistema de referencia del AAM, $\bar{z}$ son los puntos sobre la imagen, y f es el modelo de movimiento con parámetros $\bar{\mu}$. Normalmente f suele ser una transformación de semejanza en el plano cuyo vector $\bar{\mu}$ consta de cuatro parámetros (traslación, rotación y escala).

La principal utilidad de los AAM se pone de manifiesto cuando tenemos una imagen nueva, que no se encontraba entre los ejemplos, y queremos encontrar el vector de parámetros $\bar{c}$ o incluso directamente $\bar{s}$ y $\bar{g}$. A tal fin en todas las aproximaciones a los AAM se minimiza la diferencia entre los niveles de gris de una imagen sintetizada con el modelo estadístico sobre la forma media y los niveles de gris de la nueva imagen rectificada sobre la misma forma media [22]:

$$\min_{\bar{\mu},\bar{c}} ||\mathbf{I}(f(w(\bar{x}, \bar{s}), \bar{\mu}), t) - \bar{g}||^2 \quad \forall\ \bar{x} \in \bar{S}, \tag{4.60}$$

donde $\bar{g}$ y $\bar{s}$ son función de $\bar{c}$ de acuerdo con la ecuación (4.58); w es una función (afín a trozos) que toma coordenadas, $\bar{x}$, sobre el sistema de referencia de la forma media y las transforma en sus coordenadas sobre $\bar{s}$; y f es un modelo de movimiento de parámetros $\bar{\mu}$. El resultado es un algoritmo basado en descenso del gradiente que permite encontrar $\bar{c}$, $\bar{b}_s$ y $\bar{b}_g$ así como los parámetros del modelo de movimiento $\bar{\mu}$.

Existen diferentes aproximaciones en la literatura a la minimización (4.60) que permiten el funcionamiento en tiempo real [22]. La mayoría de las soluciones (aditivas) a la minimización realizan la simplificación de tomar un Jacobiano, de la imagen con respecto a los parámetros del modelo ($\bar{c}$, $\bar{\mu}$), constate para aumentar la velocidad de proceso. Esta aproximación únicamente es correcta en un entorno muy cercano a los ($\bar{c}$, $\bar{\mu}$) empleados en la construcción del Jacobiano y por eso es necesario corregir las estimaciones de los incrementos de ($\bar{c}$, $\bar{\mu}$) multiplicando por un factor corrector entre 0,25 y 1. La aproximación de Baker y Matthews [4], por su parte, se basa en el algoritmo composicional inverso de la sección 4.1.5 con el que se consigue un Jacobiano constante para calcular los parámetros del AAM y la minimización de (4.60) en tiempo real. El algoritmo de Baker y Matthews es más riguroso matemáticamente que los métodos anteriormente utilizados para minimizar (4.60) [22]. Aunque es importante recalcar que en el caso de los AAM el modelo de movimiento no forma un semigrupo, y el algoritmo presentado en [4] es una aproximación un verdadero algoritmo composicional inverso.

4.3. Conclusiones

En el contexto de la alineación con movimiento rígido el algoritmo de Lucas y Kanade no necesita más que un modelo de movimiento f derivable. En el otro plato de la balanza nos encontramos con que es mucho menos eficiente computacionalmente que el algoritmo de Baker o el de Hager. Por su parte, los dos algoritmos de

alineación incremental eficientes, el de la factorización del Jacobiano de Hager y el composicional inverso de Baker (el algoritmo de Jurie [62] es equivalente al composicional inverso de Baker), imponen severas restricciones a los modelos de movimiento. El composicional inverso exige que f forme un semigrupo y la factorización del Jacobiano impone que las derivadas de f sean factorizables. Incluso en [4, 3] se asegura que la técnica de factorización del Jacobiano no se puede aplicar con homografías, esto es, un modelo de movimiento proyectivo. En el capítulo 5 se demostrará que sí es posible el seguimiento proyectivo empleando la factorización del Jacobiano.

En cuanto a la alineación con movimiento no rígido, el algoritmo *Eigentracking* de Black y Jepson [10] (ver sección 4.2.2) construye un modelo estadístico de las variaciones en los niveles de gris debidos al movimiento no rígido. En su formulación original la minimización de (4.49) supone recalcular el Jacobiano de la imagen reconstruida con respecto a los parámetros de apariencia, $\mathbf{M}_{et}$, y por tanto $\mathbf{M}_{et}^{\top}\mathbf{M}_{et}$ en cada imagen de la secuencia lo que supone un algoritmo poco eficiente. El modelo lineal basado en PCA se puede aprender en un proceso automático en el que se estiman los parámetros de movimiento (para un modelo afín) y la base del PCA en una secuencia de imágenes [65]. Por otro lado, los AAMs presentan modelos muy compactos que se pueden emplear para el seguimiento en tiempo real. El principal problema con los AAM es la construcción del modelo estadístico de forma y textura que en la mayoría de las ocasiones supone la localización manual de los puntos de interés en cientos de imágenes. Aunque existen procedimientos automáticos [23] la localización de los puntos de interés constituye un problema difícil.

Capítulo 5

Seguidor basado en plantillas

Como ya vimos en el capítulo 4 existen básicamente dos algoritmos eficientes de alineación incremental: el composicional inverso de Baker [4] y la factorización del Jacobiano de Hager y Belhumeur [40]. Por otro lado, Hager y Belhumeur en [40] únicamente resuelven la factorización del Jacobiano para unos pocos modelos de movimiento: traslación; rotación, traslación y escala; afín y un modelo relacionado con el trabajo de Black y Yacoob [9] (ver figura 5.1).

En este capítulo se demuestra que, al contrario de lo afirmado en [3, 4], es posible factorizar el Jacobiano para seguir un plano que se deforma proyectivamente y, al mismo tiempo, estimar su posición relativa con respecto a la cámara. También se propone un procedimiento para la selección del conjunto de píxeles de la plantilla más informativos para el seguimiento. Esta técnica de selección de píxeles permitirá acelerar considerablemente el proceso de seguimiento, con una mínima repercusión en la precisión. El interés del seguidor resultante radica en que el modelo afín, resuelto por Hager y Belhumeur, no es aplicable a una cámara muy cercana al usuario (como lo es una cámara web o las de los teléfonos móviles), y sí lo es el modelo proyectivo.

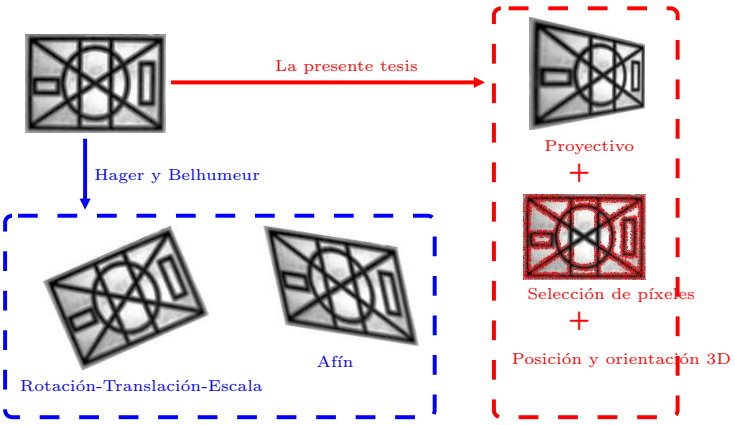


Figura 5.1: Aportaciones fundamentales: seguimiento proyectivo, estimación de la posición y orientación 3D y selección de píxeles.

5.1. Factorización del Jacobiano proyectivo

En esta sección introduciremos la factorización del Jacobiano para el modelo de movimiento proyectivo (ver sección 4.1.2). Para que un modelo f pueda ser factorizado es suficiente que éste sea lineal y es necesario que esté definida su inversa, f^{-1}. La formulación basada en coordenadas homogéneas del modelo proyectivo que veremos a continuación cumple con las dos condiciones y hace posible la factorización del Jacobiano.

Sean $\bar{x} = (u, v)^\top$ y $\bar{x}_h = (i, j, k)^\top$ respectivamente las coordenadas cartesianas y las proyectivas de un píxel entonces la relación entre ellas es la función p tal que

$$\bar{x}_h = \begin{pmatrix} i \\ j \\ k \end{pmatrix} \rightarrow \bar{x} = p(\bar{x}_h) = \begin{pmatrix} i/k \\ j/k \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} \forall \bar{x} \in I(\bar{x}) \tag{5.1}$$

La ecuación f que describe el movimiento de una región plana será entonces una trasformación lineal proyectiva en 2D,

$$f(\bar{x}_h, \bar{\mu}) = \mathbf{H} \cdot \bar{x}_h = \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & 1 \end{pmatrix} \begin{pmatrix} i \\ j \\ k \end{pmatrix}, \tag{5.2}$$

donde el vector de parámetros de movimiento ahora es $\bar{\mu} = (a, b, c, d, e, f, g, h)^\top$. Es necesario encajar las coordenadas homogéneas dentro del esquema de trabajo del algoritmo de factorización del Jacobiano y es lo que haremos en las secciones siguientes.

5.1.1. Seguimiento proyectivo

Lo primero que habrá que hacer es redefinir *la ecuación de constancia de los niveles de gris* en función de las coordenadas homogéneas de los píxeles plantilla de referencia:

$$I(p(f(\bar{x}_h, \bar{\mu}_t)), t) = I_r(p(\bar{x}_h)) = I_r(\bar{x});\ \forall \bar{x} \in \mathcal{R}. \tag{5.3}$$

donde $\bar{x}_h = (\bar{x}^\top\ 1)^\top$ son las coordenadas homogéneas correspondientes a las coordenadas cartesianas $\bar{x}$ sobre la plantilla de referencia, I_r. Por tanto la función de coste a minimizar para el caso proyectivo es

$$\min_{\bar{\mu}} \parallel \mathbf{I}(p(f(\bar{x}_h, \bar{\mu}_t + \delta\bar{\mu})), t + \delta t) - \mathbf{I}_r(p(\bar{x}_h)) \parallel^2 . \tag{5.4}$$

Desarrollando (5.4) en serie de Taylor alrededor del punto ($\bar{\mu}_t$, t) obtenemos

$$\min_{\bar{\mu}} \parallel \mathbf{M}_p \delta\bar{\mu} + \mathbf{I}(p(f(\bar{x}_h, \bar{\mu}_t)), t + \delta t) - \mathbf{I}_r(p(\bar{x}_h))) \parallel^2, \tag{5.5}$$

donde $\mathbf{M}_p$ es el Jacobiano de los niveles de gris con respecto a los parámetros de movimiento,

$$\mathbf{M}_p = \left[\frac{\partial I(p(f(\bar{x}_h, \bar{\mu}_t)), t)}{\partial \bar{\mu}} \Big|_{\bar{\mu}=\bar{\mu}_t} \right] \tag{5.6}$$

$$= \left[\frac{\partial I(p(\bar{y}_h), t)}{\partial \bar{y}_h} \Big|_{\bar{y}_h = f(\bar{x}_h, \bar{\mu}_t)} \right]^\top \left[\frac{\partial f(\bar{x}_h, \bar{\mu}), t)}{\partial \bar{\mu}} \Big|_{\bar{\mu}=\bar{\mu}_t} \right]. \tag{5.7}$$

Hasta aquí tenemos un algoritmo proyectivo con la misma carga de cómputo que el de Lucas y Kanade (ver sección 4.1.1). En la siguiente sección veremos cómo es posible factorizar el Jacobiano para el caso proyectivo.

5.1.2. Factorización del Jacobiano

Si tomamos derivadas con respecto a $\bar{x}_h$ en ambos lados de la ecuación (5.3) y despejamos obtendremos

$$\left[\frac{\partial I(p(\bar{y}_h), t)}{\partial \bar{y}_h} \Big|_{\bar{y}_h = f(\bar{x}_h, \bar{\mu}_t)} \right]^\top = \left[\frac{\partial I_r(p(\bar{x}_h))}{\partial \bar{x}_h} \right]^\top \left[\frac{\partial f(\bar{x}_h, \bar{\mu}_t)}{\partial \bar{x}_h} \right]^{-1} \tag{5.8}$$

Introduciendo (5.8) en (5.7)

$$\mathbf{M}_p = \left[\frac{\partial I_r(p(\bar{x}_h))}{\partial \bar{x}_h} \right]^\top \left[\frac{\partial f(\bar{x}_h, \bar{\mu}_t)}{\partial \bar{x}_h} \right]^{-1} \left[\frac{\partial f(\bar{x}_h, \bar{\mu})}{\partial \bar{\mu}} \Big|_{\bar{\mu}=\bar{\mu}_t} \right]. \tag{5.9}$$

Luego, el Jacobiano $\mathbf{M}_p$ depende del gradiente de la plantilla de referencia, I_r, y las derivadas del modelo de movimiento. La ecuación (5.9) define la factorización del Jacobiano para el caso proyectivo. Las derivadas de f implicadas en la factorización son fáciles de calcular:

$$\left[\frac{\partial f(\bar{x}_h, \bar{\mu}_t)}{\partial \bar{x}_h} \right]^{-1} = \mathbf{H}^{-1} \tag{5.10}$$

$$\left[\frac{\partial f(\bar{x}_h, \bar{\mu})}{\partial \bar{\mu}} \Big|_{\bar{\mu}=\bar{\mu}_t} \right] = \begin{pmatrix} i & 0 & 0 & j & 0 & 0 & k & 0 \\ 0 & i & 0 & 0 & j & 0 & 0 & k \\ 0 & 0 & i & 0 & 0 & j & 0 & 0 \end{pmatrix} \tag{5.11}$$

El problema fundamental reside ahora en el cálculo la derivada de los niveles de gris de la plantilla con respecto a las coordenadas homogéneas, $\frac{\partial I_r(p(\bar{x}_h))}{\partial \bar{x}_h}$. Si derivamos en la ecuación del gradiente con respecto a las coordenadas homogéneas obtendremos

$$\left[\frac{\partial I_r(p(\bar{x}_h))}{\partial \bar{x}_h} \right]^\top = \left[\frac{\partial I_r(\bar{x})}{\partial \bar{x}} \Big|_{\bar{x}=p(\bar{x}_h)} \right]^\top \left[\frac{\partial p(\bar{x}_h)}{\partial \bar{x}_h} \right]. \tag{5.12}$$

Es decir, el gradiente de la imagen con respecto a las coordenadas homogéneas es función del gradiente con respecto a las coordenadas cartesianas. Por otro lado tenemos que

$$\left[\frac{\partial p(\bar{x}_h)}{\partial \bar{x}_h}\right] = \begin{pmatrix} \frac{\partial u}{\partial i} & \frac{\partial u}{\partial j} & \frac{\partial u}{\partial k} \\ \frac{\partial v}{\partial i} & \frac{\partial v}{\partial j} & \frac{\partial v}{\partial k} \end{pmatrix} = \begin{pmatrix} \frac{1}{k} & 0 & -\frac{i}{k^2} \\ 0 & \frac{1}{k} & -\frac{j}{k^2} \end{pmatrix}. \tag{5.13}$$

Dado que las coordenadas homogéneas $\bar{x}_h$ con las que tratamos en la ecuación de constancia de los niveles de gris (5.3) vienen de la plantilla de referencia, tenemos que la coordenada homogénea $k = 1$. Por tanto, introduciendo la ecuación (5.13) en (5.12) tendremos que

$$\left[\frac{\partial I_r(p(\bar{x}_h))}{\partial \bar{x}_h}\right]^\top = \left(\frac{\partial I_r(\bar{x})}{\partial u}, \frac{\partial I_r(\bar{x})}{\partial v}, -\left(i\frac{\partial I_r(\bar{x})}{\partial u} + j\frac{\partial I_r(\bar{x})}{\partial v}\right)\right) \tag{5.14}$$

Por último tomando las derivadas de f en las ecuaciones (5.10) y (5.11) obtenemos la factorización

$$\begin{aligned} \left[\frac{\partial f(\bar{x}_h, \bar{\mu}_t)}{\partial \bar{x}_h}\right]^{-1} \left[\frac{\partial f(\bar{x}_h, \bar{\mu})}{\partial \bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t}\right] &= \mathbf{H}^{-1} \left(\begin{array}{c|c|c} i\mathbf{I}_{3\times 3} & j\mathbf{I}_{3\times 3} & \begin{array}{c} k\mathbf{I}_{2\times 2} \\ \hline \mathbf{0}_{1\times 2} \end{array} \end{array}\right) \\ &= \left(\begin{array}{c|c|c} i\mathbf{H}^{-1} & j\mathbf{H}^{-1} & k\mathbf{H}_{12}^{-1} \end{array}\right) \\ &= \mathbf{\Gamma}(\bar{x}_h)\mathbf{\Sigma}(\bar{\mu}), \end{aligned}$$

donde $\mathbf{H}_{12}^{-1}$ es la matriz que consta de las dos primeras columnas de $\mathbf{H}^{-1}$, $\mathbf{I}_{q\times q}$ es la matriz identidad de dimensión $q \times q$, $\mathbf{0}_{q\times q}$ es una matriz de ceros de dimensión $q \times q$ y

$$\begin{aligned} \mathbf{\Gamma}(\bar{x}_h) &= \left(\begin{array}{c|c|c} i\mathbf{I}_{3\times 3} & j\mathbf{I}_{3\times 3} & k\mathbf{I}_{3\times 3} \end{array}\right), \\ \mathbf{\Sigma}(\bar{\mu}) &= \begin{pmatrix} \mathbf{H}^{-1} & 0 & 0 \\ 0 & \mathbf{H}^{-1} & 0 \\ 0 & 0 & \mathbf{H}_{12}^{-1} \end{pmatrix}. \end{aligned}$$

En este caso la matriz $\mathbf{\Sigma}$ es 9×8 por lo que no tiene inversa. Por tanto, la ecuación para calcular $\delta\mu$ será (4.18). En este caso podremos precalcular únicamente $\mathbf{M}_0^\top \mathbf{M}_0$.

5.2. Estimación de la posición y orientación

En esta sección vamos a mostrar cómo es posible estimar la orientación y traslación de la región plana en seguimiento, P, a partir de $\mathbf{H}_r^n$ empleando un sistema de visión calibrada (o incluso autocalibración).

Hasta ahora el algoritmo de seguimiento nos permite calcular, en tiempo real, la trasformación proyectiva entre la plantilla de referencia y la imagen tomada en el instante n, $\mathbf{H}_r^n$ (ver figura 5.2). Si queremos tener información 3D tenemos que calcular 2 trasformaciones proyectivas adicionales: una desde P hasta la plantilla de

referencia, $\mathbf{H}_P^r$, y otra desde P al plano imagen en el instante n, $\mathbf{H}_P^n$. Sin pérdida de generalidad, y de cara a simplificar las ecuaciones elegimos el sistema de coordenadas de la escena de tal forma que tenga los ejes X e Y coincidentes con el plano P y el Z perpendicular al mismo (ver figura5.2).

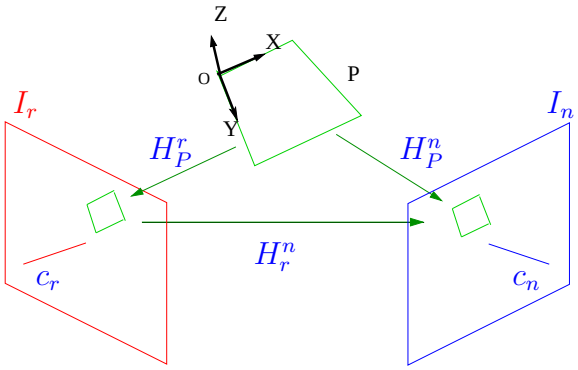


Figura 5.2: Seguimiento de planos en 3D

Trasformaciones proyectivas involucradas en el seguimiento de planos en 3D.

La trasformación proyectiva $\mathbf{H}_P^r$ se puede calcular antes de comenzar el seguimiento, empleando la proyección de al menos cuatro puntos conocidos en P. Sean $(X_P, Y_P)^\top$ las coordenadas cartesianas de un punto conocido sobre P y sean $(x_r, y_r)^\top$ las coordenadas cartesianas de la proyección de (X_P, Y_P) sobre I_r (esto es, sobre la plantilla de referencia). $\mathbf{H}_P^r$ se puede calcular a partir de:

$$\begin{pmatrix} x_r \\ y_r \\ 1 \end{pmatrix} = \mathbf{H}_P^r \begin{pmatrix} X_P \\ Y_P \\ 1 \end{pmatrix}. \tag{5.15}$$

Por otro lado, la proyección de un punto $(X_P, Y_P)^\top$ sobre I_n en el instante de tiempo n viene dada por

$$\begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} = \lambda \mathbf{K}[\ \mathbf{R}|\bar{t}\ ] \begin{pmatrix} X_P \\ Y_P \\ 0 \\ 1 \end{pmatrix}, \tag{5.16}$$

donde $\mathbf{R}$ y $\bar{t}$ son respectivamente la orientación y la posición de P en el sistema de coordenadas de la cámara, λ es un factor de escala y $\mathbf{K}$ es la matriz de intrínsecos de la cámara.

Introduciendo en (5.16) el hecho de que todos los puntos de P tienen coordenadas $Z = 0$, $\mathbf{H}_P^n$ se puede reescribir como:

$$\begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} = \lambda \mathbf{K}[\bar{r}_1\ \bar{r}_2\ \bar{t}] \begin{pmatrix} X_P \\ Y_P \\ 1 \end{pmatrix} = \mathbf{H}_P^n \begin{pmatrix} X_P \\ Y_P \\ 1 \end{pmatrix} \tag{5.17}$$

donde $\bar{r}_i$ es la i-ésima columna de la matriz $\mathbf{R}$.

Por tanto, de (5.15) y (5.17)

$$\begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} = \underbrace{\mathbf{H}_P^n(\mathbf{H}_P^r)^{-1}}_{\mathbf{H}_r^n} \begin{pmatrix} x_r \\ y_r \\ 1 \end{pmatrix} = \underbrace{\lambda\mathbf{K}[\bar{r}_1\ \bar{r}_2\ t](\mathbf{H}_P^r)^{-1}}_{\mathbf{H}_r^n} \begin{pmatrix} x_r \\ y_r \\ 1 \end{pmatrix} \tag{5.18}$$

De donde obtenemos la relación entre la transformación proyectiva $\mathbf{H}_r^n$ estimada por el seguidor con la orientación y traslación de P. Por tanto, si se conocen los intrínsecos $\mathbf{K}$ y las transformaciones proyectivas $\mathbf{H}_P^r$ y $\mathbf{H}_r^n$, podemos calcular $\mathbf{H}^*$ [82],

$$\mathbf{H}^* = \mathbf{K}^{-1}\mathbf{H}_r^n\mathbf{H}_P^r = \lambda[\bar{r}_1\ \bar{r}_2\ \bar{t}] \tag{5.19}$$

Para obtener la traslación y la rotación de $\mathbf{H}^*$ necesitamos imponer algunas restricciones:

- $\| \bar{r}_1 \| = \| \bar{r}_2 \| = 1$, De esta forma obtenemos λ y de ahí $\bar{r}_1$, $\bar{r}_2$ y $\bar{t}$.
- Ya que $\mathbf{R}$ es una matriz de rotación, sus columnas forman un sistema de referencia a derechas luego $\mathbf{R} = [\bar{r}_1\ \ \bar{r}_2\ \ \bar{r}_1 \times \bar{r}_2]$.

Por otro lado, si no se conocen los intrínsecos de la cámara, siempre podremos estimar algunos de ellos a partir de la homografía $\mathbf{H}_r^n\mathbf{H}_P^r$ en la imagen n-ésima, o todos ellos empleando conocidos algoritmos de autocalibración a partir de $k > 1$ imágenes del plano [86].

5.3. Experimentos con el seguidor proyectivo

En las secciones anteriores se ha desarrollado un algoritmo de seguimiento de planos en 3D. En esta sección pretendemos validarlo primero en el seguimiento de un plano, y después en el seguimiento del rostro humano.

Para estudiar el comportamiento del seguidor 3D sobre planos emplearemos dos secuencias de una plantilla de calibración plana. Ambas secuencias se han calibrado para obtener los intrínsecos y extrínsecos (rotación y traslación 3D) de la cámara en cada una de las imágenes. Para ello hemos empleado el paquete de calibración de cámaras para Matlab desarrollado por Jean Yves Bouguet [11] pero en su versión modificada por Vezhnevets Vladimir [99] para la extracción automática de las esquinas de la plantilla. El resultado son sendas secuencias de imágenes con un plano en movimiento del que conocemos una estimación de su orientación y posición con respecto a la cámara (los extrínsecos).

Una de las hipótesis de partida de la alineación incremental de imágenes es que entre una imagen y la siguiente el movimiento del objeto es pequeño (reflejado en la aproximación de primer orden). Dado que no se pretende limitar la velocidad de movimiento del objetivo, hay que desarrollar algún método que permita continuar

el seguimiento a pesar de que se produzcan grandes desplazamientos. La solución a este problema consiste en una pirámide gaussiana con diferentes niveles de resolución tanto para la plantilla como para las imágenes rectificadas. De esta forma se comienza iterando en el nivel de resolución más bajo (lo que permite enfrentarse a desplazamientos mayores) para ir aumentando paulatinamente la resolución hasta obtener el incremento de parámetros de movimiento. En el primer experimento emplearemos una secuencia de 250 imágenes de 320×240 píxeles (ver figura 5.3) para estudiar el efecto del número de niveles de resolución y de iteraciones por nivel.

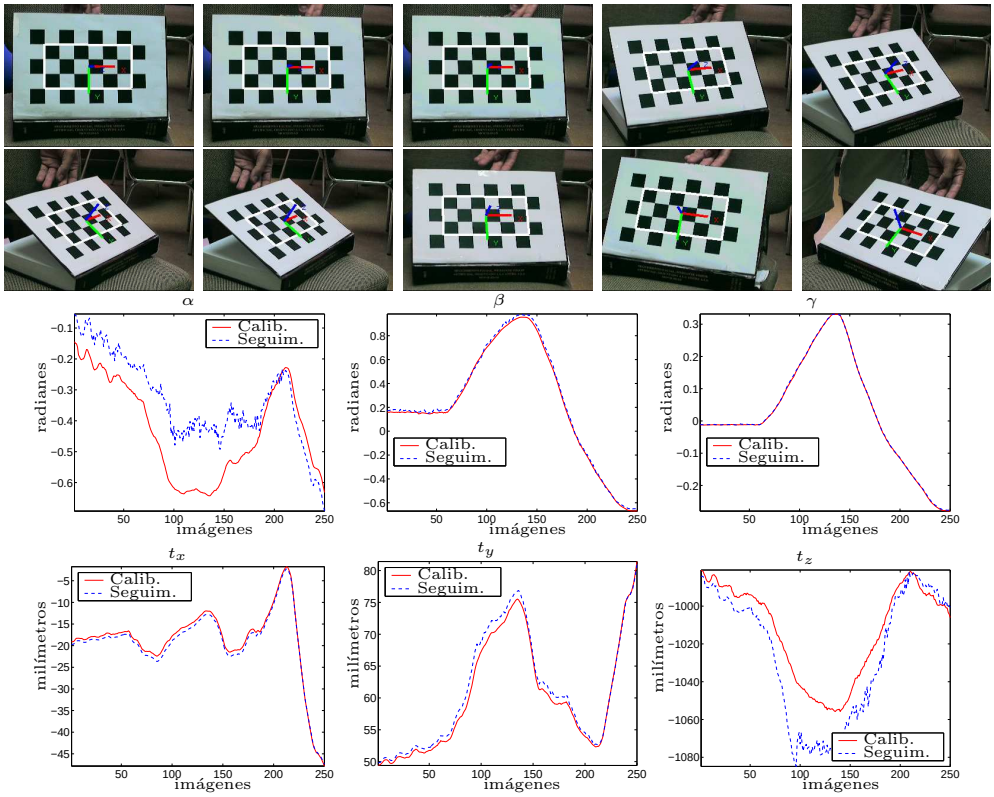


Figura 5.3: Resultados sobre la primera secuencia de imágenes de un objeto plano: 320×240 píxeles por imagen, 2 niveles de resolución y 3 iteraciones por nivel. En las dos primeras filas se muestran las imágenes 1, 20, 40, 80, 100, 120, 140, 180, 200 y 250. Sobre ellas aparecen la estimación de posición y orientación del plano. En la tercera fila se muestra la estimación de cada uno de los ángulos en el seguimiento frente a los valores provenientes de la calibración. En la última fila se muestra la estimación de la traslación en cada uno de los ejes frente a los valores provenientes de la calibración.

Si nos fijamos en cualquiera de las gráficas de la figura 5.3, observaremos que los errores en la estimación de la posición y orientación 3D, como era de esperar, son mayores o menores dependiendo de la orientación y de la distancia del plano a la cámara. En la estimación de la rotación alrededor del eje horizontal de la cámara, α, el error es de hasta 12 grados entre las imágenes 100 y 150, fundamentalmente

debido a problemas de *aliasing* en el rectificado de la plantilla cuando la orientación del plano es de 40 grados tanto alrededor del eje horizontal como el vertical. Sin embargo, en cuanto la orientación del plano se acerca a la frontal a la cámara el error en la estimación de α se reduce prácticamente a cero. En cambio, en la estimación de la distancia del plano a la cámara, t_z, tenemos un error máximo de 4,5 centímetros, lo que supone únicamente el 4 % de la distancia a la que se encuentra el plano (1 metro y 4 centímetros).

Pretendemos establecer la bondad del seguimiento basado en niveles de gris comparándolo con los extrínsecos provenientes de la calibración de la secuencia. Estudiaremos el error cuadrático medio entre la posición de cuatro esquinas de la plantilla de seguimiento estimada por nuestro seguidor y la posición de las mismas de acuerdo con los extrínsecos de la calibración (proyectando las esquinas 3D sobre la imagen). En la figura 5.4 se muestran los resultados empleando una plantilla de seguimiento de 158×79 píxeles sobre la secuencia de la figura 5.3.

Los mejores resultados los obtenemos con dos niveles de resolución con errores por debajo de los 0,73 píxeles (precisión subpíxel). El hecho de realizar un máximo de 6 iteraciones por nivel de resolución frente a 3, no implica más que reducir el error en 0,03 píxeles (de 0,73 a 0,7). Cabe resaltar el que con tres niveles de resolución obtengamos un error máximo de 50 píxeles, esto es debido a que con una resolución tan pequeña no tenemos suficiente información para resolver la ecuación (4.18). La conclusión es que empleando dos niveles de resolución y un máximo de 3 iteraciones por nivel tendremos unos buenos resultados en el seguimiento.

Con el método de Hager y Belhumeur el Jacobiano, $\mathbf{M}_p$, se calcula en función de los gradientes de la plantilla de seguimiento. Por tanto, si conseguimos una plantilla sin ruido en los niveles de gris, por ejemplo mediante el promedio de varias imágenes, tendremos un Jacobiano independiente del ruido [3]. Sin embargo, aún nos queda una fuente de ruido que puede afectar a nuestro algoritmo de seguimiento: el proveniente del proceso de adquisición de las imágenes. En el segundo experimento emplearemos una secuencia de 150 imágenes de 640×480 píxeles (ver figura 5.5) para estudiar el efecto del ruido en los niveles de gris.

Hemos realizado el seguimiento de la plantilla de calibración sobre la secuencia del segundo experimento añadiendo ruido gaussiano en los niveles de gris. Dado que se trata de imágenes en color se ha supuesto que el ruido en cada canal (R, G y B) es independiente de los demás y se puede modelar como una normal de desviación típica conocida y media cero. Compararemos la posición de las cuatro esquinas de la plantilla sobre cada imagen, resultado de proyectar los puntos 3D con los extrínsecos de la calibración, con la estimada por nuestro algoritmo. Para cada una de las desviaciones típicas empleadas para generar el ruido, se han realizado 15 experimentos y se han promediado los resultados. En la figura 5.6 podemos ver la media de las distancias de las cuatro esquinas para los diferentes niveles de ruido. Según se incrementa el ruido vemos que pasamos de un error de menos de 0,66 píxeles a un error de hasta 0.82 píxeles. Es necesario un ruido con una desviación típica de 100 niveles de gris para que el error pase de 0,72 a 0.82 píxeles, o lo que es lo mismo la calidad del seguimiento se degrada pero a pesar de todo continúa. Esto

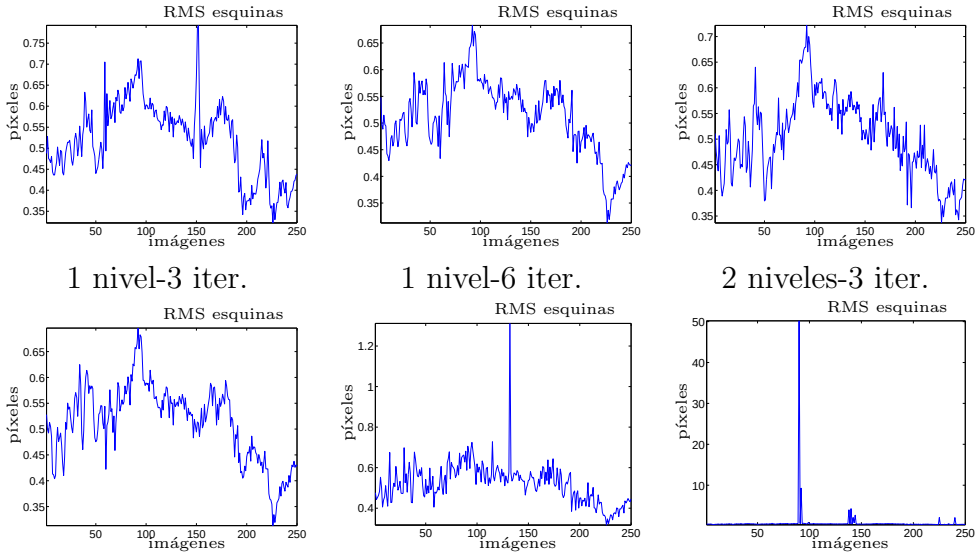


Figura 5.4: RMS de las distancias euclídeas de cada esquina del plano con respecto a la verdadera en cada imagen. Queremos observar la influencia de los niveles de resolución y el número máximo de iteraciones por nivel en el resultado final. De izquierda a derecha y de arriba a abajo se muestran los resultados para 1 nivel-3 iteraciones; 1 nivel-6 iteraciones; 2 niveles-3 iteraciones; 2-niveles-6 iteraciones; 3 niveles-3 iteraciones y 3 niveles-4 iteraciones.

es, con un número suficiente de píxeles en la plantilla de seguimiento (en este caso 158×79 píxeles) el ruido en la adquisición de las imágenes tiene que ser muy grande para que el seguimiento no pueda continuar.

Hasta el momento hemos probado el seguidor proyectivo con superficies planas, y dado que lo utilizaremos para la localización de la cara, en lo que resta de sección presentaremos resultados sobre el rostro humano. La restricción más fuerte que vamos a adoptar consiste en suponer que la superficie de la cara es aproximadamente plana. En vista de que realmente no estamos tratando con un plano (ver figura 5.7) es de suponer que los resultados para las rotaciones fuera de la frontal a la cámara (cuando la nariz oculta parte de la cara) no serán tan buenos como los obtenidos con una superficie plana.

Para probar el funcionamiento del algoritmo de seguimiento se han empleado las secuencias de Marco La Cascia [64], para las que se dispone de la posición y orientación de la cabeza obtenidos mediante un dispositivo de seguimiento magnético. En la primera imagen de cada secuencia el sistema de referencia de la cámara y del dispositivo magnético poseen la misma orientación, aunque sus orígenes se encuentran en posiciones diferentes. Para calcular la posición y orientación 3D a partir de la homografía necesitamos la matriz de intrínsecos y no se dispone de ella en las secuencias de prueba. Para solventar este problema hemos autocalibrado los intrínsecos a partir de la posición de las cuatro esquinas de la cara en una secuen-

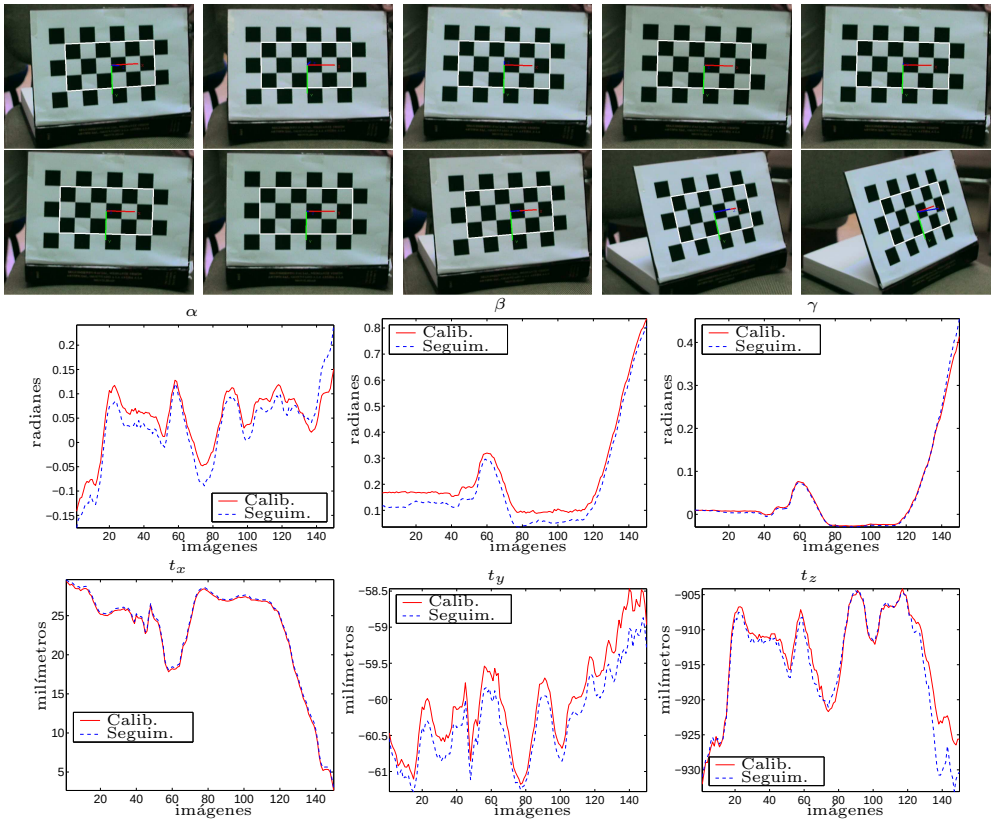


Figura 5.5: Resultados sobre la segunda secuencia de imágenes (640×480 píxeles por imagen, 1 nivel de resolución y 10 iteraciones por nivel) de un objeto plano. En las dos primeras filas se muestran una de cada 10 imágenes (a partir de la 60) y sobre ellas las estimación de posición y orientación del plano. En la tercera fila se muestra la estimación de cada uno de los ángulos en el seguimiento frente a los valores provenientes de la calibración. En la última fila se muestra la estimación de la traslación en cada uno de los ejes frente a los valores provenientes de la calibración. En cada una de las gráficas tenemos el promedio de 15 experimentos en los que se añade ruido con una desviación típica de 20 niveles de gris y media cero.

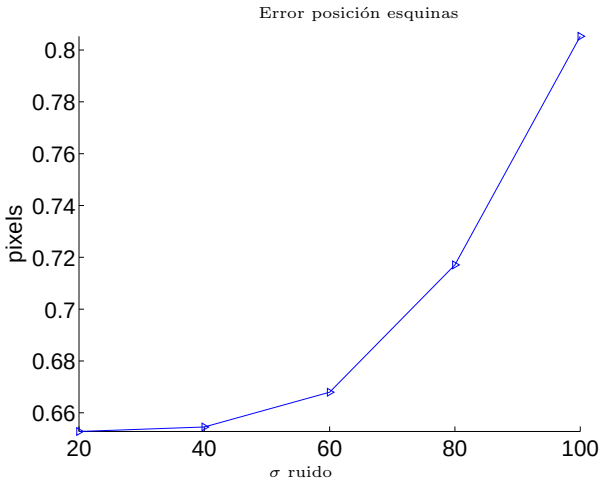


Figura 5.6: Media de la distancia euclídea de cada esquina del plano con respecto a la verdadera en cada imagen. Se añadió ruido gaussiano a cada una de las imágenes de la secuencia pruebas. Se muestran los resultados con una desviación típica de 20, 40, 60, 80 y 100 niveles de gris. En todos los casos hemos realizado 15 repeticiones de cada experimento generando ruido para las imágenes y promediando el resultado.

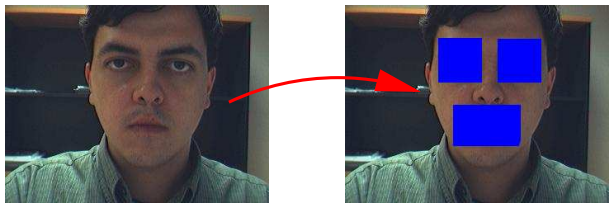


Figura 5.7: Regiones planas asociadas con el rostro humano.

cia con movimientos sencillos (no demasiado fuera de la frontal a la cámara) para nuestro seguidor, empleando el software de calibración de cámaras desarrollado por Jean Yves Bouguet [11]. En todas las gráficas que se mostrarán a continuación, y dado que el dispositivo magnético nos ofrece mediciones relativas al instante inicial, se han modificado los resultados del seguimiento para que comiencen en cero grados (y poder realizar la comparación). Por otro lado, no se emplean los píxeles del borde de las plantillas de seguimiento (ver figura 5.8) para intentar evitar los problemas con los giros fuera del plano.

Figura 5.8: Plantilla de seguimiento para la cara. La segunda imagen (la parte blanca) define los píxeles efectivamente empleados en el seguimiento.

.

En la secuencia del experimento (ver figura 5.9) los movimientos del usuario básicamente son traslaciones y rotaciones en el plano imagen (alrededor del eje Z). Las rotaciones alrededor del eje Z (cabeza hacia los hombros), o ángulo γ, se estiman casi perfectamente. Sin embargo, las rotaciones alrededor del eje X (gesto de afirmación), o ángulo α, y alrededor del eje Y (gesto de negación), o ángulo β, presentan un error de hasta 4,5 grados.

Con nuestro seguidor es posible estimar las rotaciones alrededor del eje axial de la cámara con bastante fiabilidad. Podemos estimar rotaciones alrededor del eje vertical a la cámara de hasta 10 grados (ver figura. 5.9). Más allá pueden aparecer partes del fondo en la imagen rectificada, a pesar de excluir los píxeles del borde de la plantilla. La estimación de la rotación alrededor del eje horizontal a la cámara es la menos precisa. Esto es así en parte por la aparición del fondo en la imagen rectificada y en parte porque la aparición de la nariz viola la asunción de planaridad. Por otro lado, dado que la matriz de intrínsecos es sólo aproximada, cabe la posibilidad de que el margen error sea mejorable, pero de todas formas los análisis cualitativos siguen siendo válidos. Por último, hay que destacar que las secuencias de imágenes utilizadas en las pruebas (ver [64]) se encuentran comprimidas y no ofrecen una calidad óptima para nuestros experimentos. Lo que implica, teniendo en cuenta que la plantilla de seguimiento se ha obtenido de la primera imagen de la secuencia, que no representan el caso más favorable para nuestro algoritmo.

En cualquier caso, para la cuantificación de expresiones faciales, no es tan importante la precisión en la estimación de la orientación de la cabeza como lo es la localización en la imagen de las zonas de interés del rostro. Nuestro algoritmo de seguimiento sí que permite la localización precisa de la cara en el plano imagen empleando una cámara cercana al usuario.

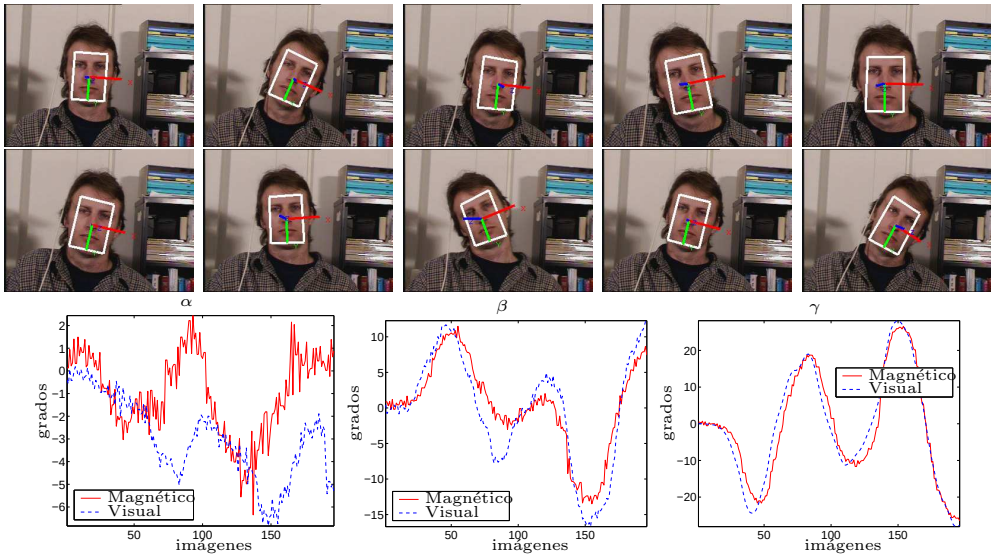


Figura 5.9: Resultados procesando la secuencia de 200 imágenes y denominada jam1. En las primeras dos filas se muestran una de cada veinte imágenes procesadas comenzando en la imagen número 20. En la tercera fila se muestran las orientaciones estimadas por el seguidor plano frente a las obtenidas por un dispositivo de posicionamiento magnético.

5.4. Selección de píxeles de la plantilla

Sólo las áreas de la imagen con alto contraste aportan información acerca del movimiento de la plantilla (ver figura 5.10, sólo los píxeles en los bordes en la imagen de la plantilla de seguimiento aportan información para el seguimiento). Si usamos todos los píxeles de la plantilla en la ecuación (4.18), la mayor parte del esfuerzo computacional se dedicará a píxeles no informativos. En esta sección aumentaremos

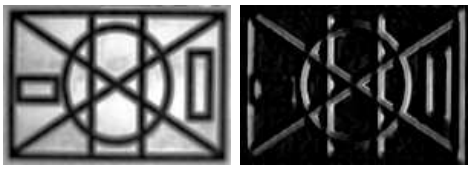


Figura 5.10: Imágenes de una plantilla de seguimiento (izquierda) y de $I(f(\bar{x}, \bar{\mu}_t), t+\delta t) - I_r(\bar{x})$ (derecha), donde el parámetro $\bar{\mu}$ representa desplazamiento horizontal. En la imagen de la derecha el nivel de gris de los píxeles es proporcional a la cantidad de información que proporcionan.

la velocidad de procesamiento del algoritmo de seguimiento reduciendo el número de píxeles de la plantilla empleados en la solución de la ecuación (4.18). Esta mejora proviene fundamentalmente de tener una matriz $\mathbf{M}_0$ más pequeña.

La matriz Jacobiana $\mathbf{M}$ de la imagen I con respecto a los parámetros de movi-

miento se puede expresar como:

$$\mathbf{M} = (\mathbf{I}_{\bar{\mu}_1}, \mathbf{I}_{\bar{\mu}_2}, \cdots, \mathbf{I}_{\bar{\mu}_n}), \tag{5.20}$$

donde $\mathbf{I}_{\bar{\mu}_i} = \frac{\partial I(\bar{x},\bar{\mu})}{\partial \bar{\mu}_i}$ es un vector columna con una entrada para cada píxel en I. Representa los cambios en los niveles de gris de una imagen inducidos por el movimiento $\bar{\mu}_i$ (ver figura 5.11). Nótese que $\mathbf{M}$ relaciona variaciones en los parámetros de movimiento con variaciones en los valores de gris. Obsérvese que la ecuación (4.18) trabaja en la dirección opuesta, empleando $\mathbf{M}$ para calcular el movimiento a partir de los cambios en los valores de gris.

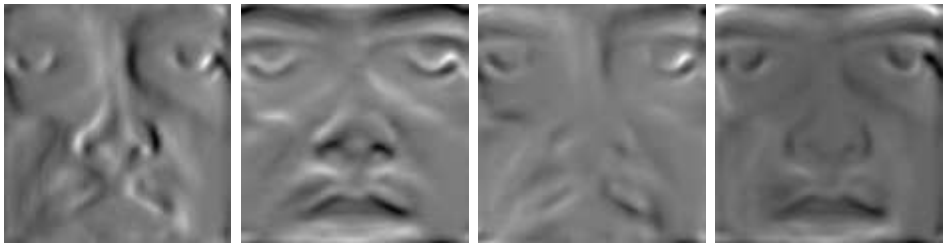


Figura 5.11: Matriz Jacobiana para una modelo de movimiento con traslación (x, y), rotación (θ) y escala (s). De izquierda a derecha, cada imagen representa respectivamente $\mathbf{I}_x$, $\mathbf{I}_y$, $\mathbf{I}_\theta$, $\mathbf{I}_s$.

Sea $\mathbf{I}_{\bar{\mu}}^{\top}(\bar{x})$ la fila de $\mathbf{M}$ correspondiente al píxel $I(\bar{x})$. Cada entrada en la fila es la derivada del píxel $I(\bar{x})$ con respecto al parámetro $\bar{\mu}_i$ $(\forall i = 1 \ldots n)$. En una fila de $\mathbf{M}$ correspondiente al píxel $\bar{x}$ tendremos las contribuciones de ese píxel al cálculo de cada uno de los parámetros de movimiento. Basta con que $\bar{x}$ contribuya fuertemente al cálculo de uno de los parámetros $\bar{\mu}_i$ (el valor absoluto de $\mathbf{I}_{\bar{\mu}_i}(\bar{x})$ sea alto) para que lo consideremos un buen píxel. Luego, un píxel con una norma $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$ pequeña no proporciona casi información para resolver (4.18). Sin embargo, un buen píxel para el seguimiento es uno con una norma $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$ grande. Por otro lado, dados dos píxeles de $I(\bar{x}_1)$ y $I(\bar{x}_2)$, uno de ellos es redundante si $\mathbf{I}_{\bar{\mu}}(\bar{x}_1) \approx \mathbf{I}_{\bar{\mu}}(\bar{x}_2)$.

Un buen conjunto de píxeles para el seguimiento es uno tal que $\mathbf{M}^{\top}\mathbf{M}$ es no singular, o lo que es lo mismo, $\mathbf{M}$ es de rango completo. Elegir el "mejor" conjunto de m píxeles es un problema de búsqueda combinatorio, ya que todos los $\binom{m}{N}$ conjuntos de píxeles deberían considerarse para elegir el más informativo. En el contexto del registrado de imágenes, Dellaert selecciona m píxeles aleatoriamente del 20 % con la norma $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$ más alta [27]. En lo que sigue presentaremos dos procedimientos de selección de píxeles y observaremos su comportamiento experimentalmente.

5.4.1. El cierre convexo del Jacobiano

Si consideramos cada fila de la matriz Jacobiana, $\mathbf{I}_{\bar{\mu}}^{\top}(\bar{x})$, como un punto en un espacio n-dimensional, entonces, los puntos en el cierre convexo de esta nube son aquellos con el la mayor $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$ y la menor redundancia. Llamemos a este conjunto

de puntos la *nube Jacobiana*. Calcular el cierre convexo de una nube Jacobiana con miles de puntos en un espacio n-dimensional (con n=4 en el caso rotación-traslación escala, n=6 en el caso afín y n=8 en el caso proyectivo) puede llegar a ser muy costoso en tiempo.

Como se puede observar en la figura 5.12 (izquierda), la distribución de los puntos para el modelo rotación-traslación-escala (en adelante RTE) se encuentra altamente correlada, con dos direcciones espaciales representando el 99.99 % de la varianza total de la nube. En el caso del modelo afín (ver figura 5.12, centro) son cuatro las direcciones espaciales las que representan el 90.99 % de la varianza total, mientras que para el modelo proyectivo tenemos de nuevo dos direcciones espaciales representando el 99.99 % de la varianza total (ver figura 5.12, derecha). Una buena aproximación al cierre convexo de la nube sería calcular el cierre convexo de su proyección en las direcciones principales. En el caso de los modelos de movimiento RTE y proyectivo el espacio resultante de la proyección (espacio de componentes principales) es bidimensional luego, por facilidad de visualización, nos restringiremos a ellos al estudiar la elección de los píxeles. En ambos casos $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$ se ve reflejado en la distancia de un punto en el espacio de componentes principales al origen de coordenadas (ver figura 5.13) y la dirección de $\mathbf{I}_{\bar{\mu}}(\bar{x})$, asociada al tipo de borde, se pondrá de manifiesto en la dirección del vector que une el punto en este espacio bidimensional con el origen de coordenadas.

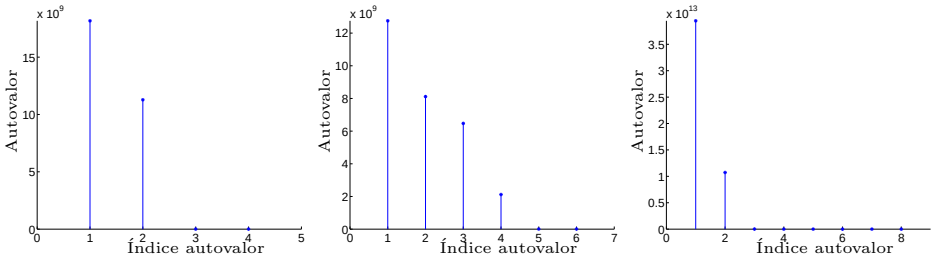


Figura 5.12: Autovalores de la matriz de covarianzas de la nube Jacobiana de los modelos de movimiento: rotación-traslación-escala (izquierda), afín (centro) y proyectivo (derecha). En este caso se han calculado los autovalores correspondientes a la imagen izquierda de la figura 5.10.

Si observamos las figuras 5.13, 5.14 y 5.15 podemos extraer dos propiedades de $\mathbf{I}_{\bar{\mu}}(\bar{x})$ que nos van a ayudar a elegir los mejores píxeles de la plantilla de seguimiento:

- La posición de $\mathbf{I}_{\bar{\mu}}^{\top}(\bar{x})$ en el espacio n-dimensional nos indica el *tipo de borde* que aparece en el píxel de coordenadas $\bar{x}$ en la plantilla de referencia. El tipo de borde depende tanto de la forma (una esquina, una línea recta, una zona con poco contraste, etc) como de su orientación (no será lo mismo una línea horizontal que una vertical o la esquina superior derecha que la esquina superior izquierda).

- El valor de $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$ indica la magnitud de la variación en los valores de gris

debido una variación de $\bar{\mu}$. Esa variación tiene su origen, por un lado en lo fuerte que sea el borde (no es lo mismo una transición del blanco al negro abrupta que una transición suave) y por otro, de su posición en la plantilla (en una rotación alrededor del centro de la plantilla, los bordes más alejados del centro se ven más afectados que los menos alejados y, por tanto la variación en los niveles de gris será mayor en los más exteriores).

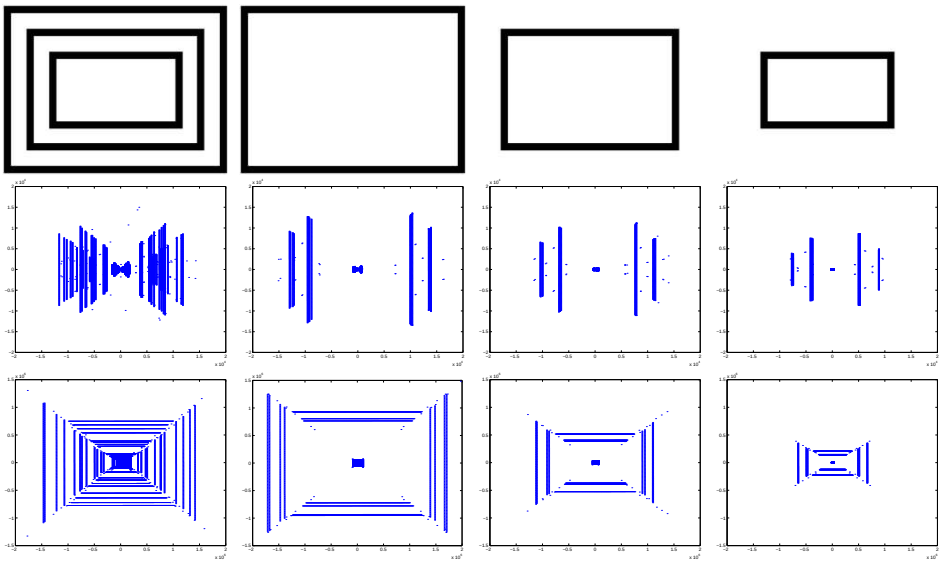


Figura 5.13: Plantilla de seguimiento y subplantillas (fila de arriba) junto con el espacio de PCA correspondiente para el modelo RTE (segunda fila) y para el modelo proyectivo (fila de abajo).

Eligiendo los puntos del cierre convexo de la nube Jacobiana vamos a obtener un conjunto de píxeles con el mayor valor de $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$ y poca redundancia. Calcularemos todos los cierres convexos de la nube Jacobiana actuando como si pelásemos una naranja (eliminando de la nube los puntos del cierre convexo antes de calcular el siguiente). Estudiaremos dos alternativas para la elección de los cierres convexos. En la primera, elegiremos todos los píxeles de un conjunto de cierres convexos seleccionados desde más externos a menos externos hasta alcanzar el número deseado de píxeles, m. En la segunda alternativa, elegiremos aleatoriamente cierres convexos de entre el 70 % de los más externos hasta alcanzar el número deseado de píxeles, m.

5.4.2. Distribución espacial uniforme

La heurística de selección del conjunto de píxeles que plantearemos en esta sección intenta conseguir la distribución espacial más uniforme posible de los píxeles sobre la plantilla de seguimiento. Para asegurarnos de tomar píxeles en todas las zonas de la imagen, colocaremos una rejilla de $f \times c$ zonas sobre la imagen. A con-

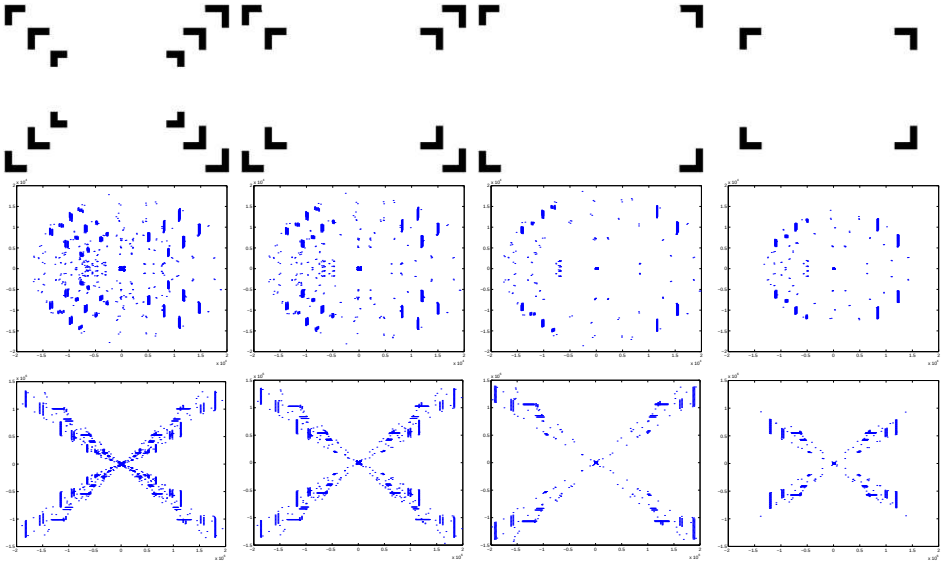


Figura 5.14: Plantilla de seguimiento y subplantillas (fila de arriba) junto con el espacio de PCA correspondiente para el modelo RTE (segunda fila) y para el modelo proyectivo (fila de abajo).

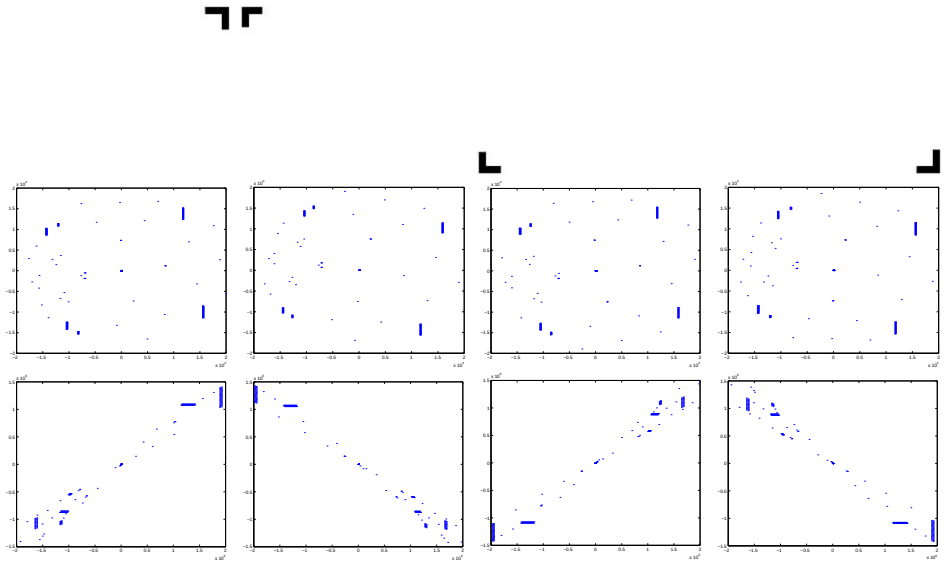


Figura 5.15: Plantilla de seguimiento y subplantillas (fila de arriba) junto con el espacio de PCA correspondiente para el modelo RTE (segunda fila) y para el modelo proyectivo (fila de abajo).

tinuación elegiremos píxeles aleatoriamente dentro de cada celda de la rejilla. El algoritmo que seguiremos será el siguiente:

1. Ordenar los píxeles en cada celda de la rejilla de mayor a menor $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$.
2. Elegir aleatoriamente $m/(f \times c)$ píxeles de entre el $p\,\%$ de los de mayor $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$ en cada una de las celdas de la rejilla.

Lo que planteamos con este algoritmo es imponer la elección de píxeles en todas y cada una de las celdas en las que se divide la imagen. De esta forma, tenemos el conjunto de píxeles que se utilizarán en el seguimiento efectivamente distribuidos por toda la plantilla de referencia.

5.5. Experimentos con la selección de píxeles

En esta sección vamos a exponer los resultados de los experimentos realizados en la selección de píxeles de la plantilla de seguimiento. Utilizaremos siempre el modelo de movimiento proyectivo y lo aplicaremos a diferentes plantillas de seguimiento. En el caso proyectivo $\mathbf{M}(\bar{\mu}_I) = \mathbf{M}_0$ así que utilizaremos como Jacobiano a $\mathbf{M}_0$ en todos los experimentos. En las diferentes pruebas realizadas compararemos cuatro variantes de las heurísticas de selección expuestas en las secciones anteriores:

- **Normas máximas aleatorias**. Esta heurística consiste en seleccionar aleatoriamente m píxeles de entre el $80\,\%$ de píxeles con mayor $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$.
- **Distribución espacial**. Esta es la heurística explicada en la sección 5.4.2.
- **Cierres convexos**. Con este nombre nos referiremos al procedimiento de selección basado en la elección de los píxeles contenidos en los cierres convexos más externos. Se toman todos los píxeles del cierre más externo, si no tenemos ya los m píxeles buscados, se eligen los que se encuentran sobre el siguiente más externo y así sucesivamente hasta obtener m píxeles o más (ver sección 5.4.1).
- **Cierres convexos aleatorio**. El último método consistirá en elegir aleatoriamente cierres convexos de entre el $70\,\%$ de los más externos, tomando todos los píxeles en cada cierre hasta obtener m o más píxeles (ver sección 5.4.1).

En nuestra primera prueba utilizaremos la misma secuencia de imágenes del primer experimento de la sección 5.3 (ver figura 5.3) y la misma plantilla de seguimiento. En este caso realizaremos 10 iteraciones (máximo) por nivel y emplearemos únicamente 1 nivel de resolución. Con esta plantilla de seguimiento también ocurre que, tanto para el modelo de movimiento rotación-traslación-escala como para el proyectivo, podemos proyectar todos los puntos de la nube Jacobiana sobre 2 dimensiones dado que dos de los autovectores se llevan el $99\,\%$ de la variabilidad (ver

figura 5.12). Para establecer cual de las cuatro heurísticas es mejor compararemos el comportamiento de las cuatro con diferente número de píxeles, m. En la figura 5.16 podemos observar la proyección de la nube Jacobiana sobre 2 dimensiones y la elección de cierres convexos en sus dos versiones.

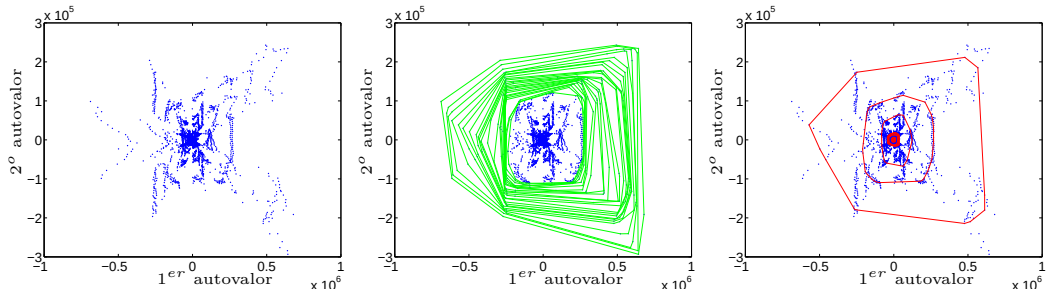


Figura 5.16: Resultados del cierre convexo proyectivo de la plantilla de calibración. A la izquierda aparece el resultado de proyectar la nube jacobiana del modelo de movimiento proyectivo en dos dimensiones. En el centro los 25 cierres convexos más alejados del centro de la nube. A la derecha aparecen 25 cierres convexos elegidos aleatoriamente de entre el 70 % de los más alejados del centro de la nube jacobiana.

En la figura 5.17 aparecen los píxeles elegidos sobre la plantilla de calibración con diferentes valores de m y empleando las cuatro heurísticas de selección. El método que resulta en una distribución menos uniforme sobre la imagen es el del cierre convexo. Tomar los píxeles sobre los cierres convexos más externos de la nube Jacobiana implica seleccionar los que se encuentran sobre los bordes más marcados de la imagen y que aportan mayor cantidad de información al seguimiento (esa es la razón por la que no se eligen píxeles en los dos cuadrados centrales de la plantilla cuando el número de cierres convexos elegidos es pequeño). Sin embargo, también esa es la causa por la que no se distribuyen uniformemente los píxeles por la plantilla.

Para establecer la calidad del seguimiento observaremos la media de las distancias entre la posición de las cuatro esquinas de la región de referencia, sobre cada imagen, estimadas por el algoritmo de seguimiento y las provenientes de la calibración. En la figura 5.18 aparecen los resultados para distintos valores de m con cada una de las heurísticas de selección. Con todas las heurísticas se produce una disminución del error en las esquinas cuando pasamos de elegir menos de 1000 píxeles ($\approx$ 8 % del número total de píxeles) a elegir alrededor de 3000 ($\approx$ 24 % del número total de píxeles). A partir de 3000, y hasta el número total de píxeles de la plantilla, el error se estabiliza en un valor que es más pequeño que el logrado con un valor de m por debajo de 3000. Esto es así porque el con 8 % de los píxeles de la plantilla, se obtiene una matriz $\mathbf{M}_0^\top \mathbf{M}_0$ con un condicionamiento peor que la que se obtiene con un 24 % (o más). Máxime cuando se trata de una plantilla de seguimiento que no es demasiado buena para el alineamiento incremental de imágenes al constar de bordes muy abruptos que no ocupan apenas superficie en en la plantilla (un tablero de ajedrez). Por otro lado, el menor error (alrededor de 0,5 píxeles) se produce con la heurística de la distribución espacial. Podemos concluir que con el seguimiento pro-

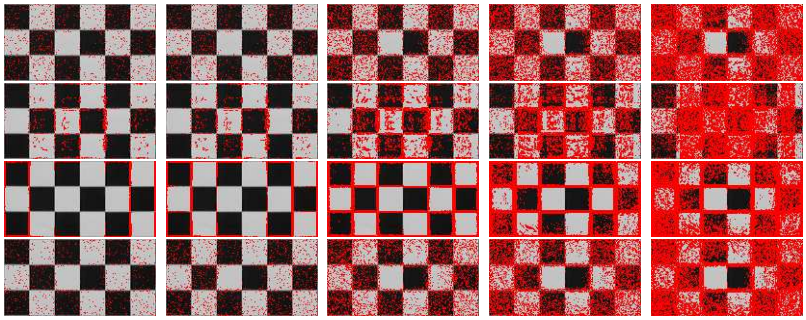


Figura 5.17: Píxeles empleados en el primer experimento de selección marcados en rojo para el modelo de movimiento proyectivo. En la primera fila se muestra la selección con el método de las normas máximas aleatorias para (de izquierda a derecha): 800, 1000, 3000, 5000 y 8000. En la segunda fila se muestra la selección con el método de dispersión espacial para (de izquierda a derecha): 853, 1001, 3042, 5064 y 8088.En la tercera fila se muestra la selección con el método del cierre convexo para (de izquierda a derecha): 892 (50 cierres convexos), 1457 (75 cierres convexos), 3075 (150 cierres convexos), 5247 (225 cierres convexos) y 7834 (300 cierres convexos). En la última fila se muestra la selección con la elección aleatoria de los cierres convexos (de izquierda a derecha): 715 (25 cierres convexos), 1369 (50 cierres convexos), 3350 (125 cierres convexos), 5288 (200 cierres convexos) y 8017 (303 cierres convexos).

yectivo y la plantilla de calibración (como un tablero de ajedrez) la mejor heurística de selección de píxeles es la distribución espacial, aunque con una diferencia mínima en precisión con las demás (0,05 píxeles). En cuanto a los tiempos de procesamiento, tenemos que con un 24 % de los píxeles de la plantilla se necesitan 7 milisegundos por imagen (recordemos que hacemos un máximo de 10 iteraciones por nivel de resolución), mientras que con todos los píxeles de la misma se necesitan 16 milisegundos por imagen. O lo que es lo mismo, obtenemos una reducción del 57 % en tiempo de cómputo con un 24 % de los píxeles de la plantilla con una pérdida mínima en precisión (0,5 píxeles de media por esquina de la región de interés).

Nuestra segunda prueba de selección de píxeles de la plantilla utiliza la misma secuencia que en el segundo experimento de la sección 5.3 (ver figura 5.5) y la misma plantilla de seguimiento. En este caso haremos 10 iteraciones por nivel de resolución (máximo) y utilizaremos 1 nivel de resolución. En este experimento pretendemos averiguar el comportamiento del seguidor y de las heurísticas de selección de píxeles en presencia de ruido gaussiano en los niveles de gris. Para ello añadimos ruido a los tres canales de color de gaussianas independientes pero con la misma desviación típica y repetimos el experimento 15 veces promediando el resultado. Tal y como podemos observar en la figura 5.19 en presencia de ruido los mejores resultados los obtenemos para el método de los cierres convexos con un error siempre por debajo de 0,7 píxeles. Esto no es sorprendente ya que empleamos el conjunto de píxeles que más información aportan al seguimiento.

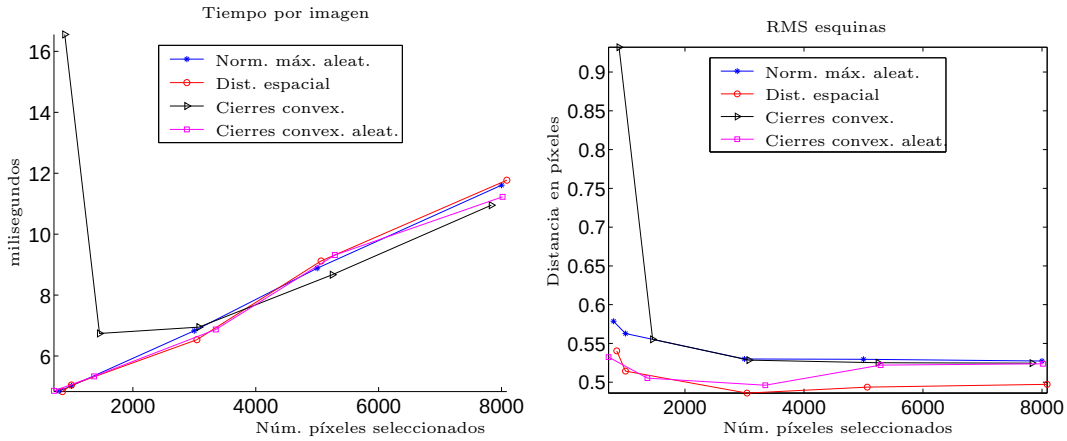


Figura 5.18: Resultados del primer experimento de selección de píxeles. De izquierda a derecha, gráfica de tiempos y error en las posición estimada de las cuatro esquinas de la región de interés sobre la imagen, empleando diferente número de píxeles para el seguimiento. En todos los casos se realizaron un máximo de 10 iteraciones del algoritmo en el un único nivel de resolución utilizado.

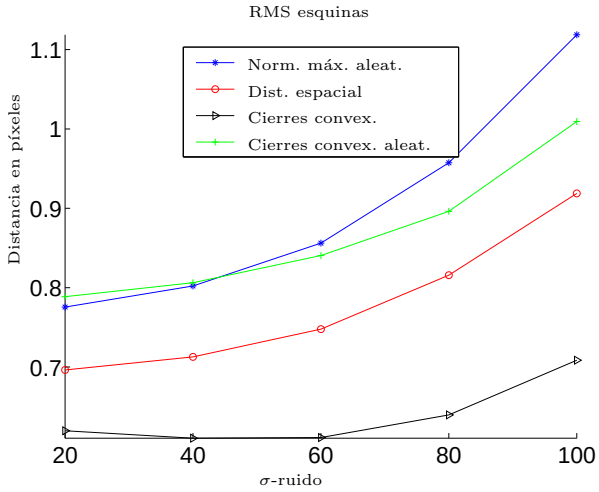


Figura 5.19: Resultados para el segundo experimento de selección de píxeles. Error en la posición estimada de las cuatro esquinas de la región de interés sobre la imagen con diferentes niveles de ruido gaussiano. En el caso de las normas máximas aleatorias empleamos 1000 píxeles, 1001 con la distribución espacial, 1457 con el de los cierres convexos y 1369 con el de los cierres convexos aleatorios para todos los niveles de ruido. En todos los casos se realizaron un máximo de 10 iteraciones en el único nivel de resolución utilizado. El resultado mostrado es el promedio de 15 repeticiones del experimento.

Con el tercer experimento pretendemos establecer el comportamiento de los métodos de selección de píxeles con una plantilla de seguimiento diferente (una vez más de un objeto plano). La plantilla que hemos utilizado ha sido diseñada por nosotros (se puede ver en la figura 5.20) con unas dimensiones de 123×80 píxeles. La distribución menos uniforme sobre la plantilla de los píxeles seleccionados (ver figura 5.20), se produce con el método de los cierres convexos. Los puntos elegidos se agrupan alrededor de los bordes con píxeles con mayor $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$, lo que se corresponde con los más exteriores de la plantilla de seguimiento (ver la elección de 25 cierres convexos en la figura 5.20). En la figura 5.21 podemos hacernos una idea de cómo funciona el seguidor proyectivo en este caso.

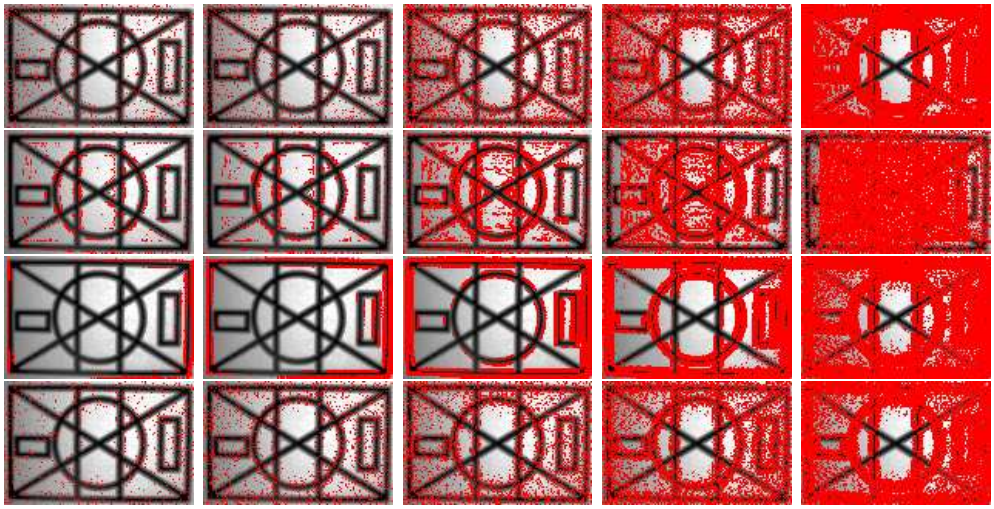


Figura 5.20: Píxeles empleados en el tercer experimento de selección marcados en rojo para el modelo de movimiento proyectivo. En la primera fila se muestra la selección con el método de las normas máximas aleatorias para (de izquierda a derecha): 800, 1000, 3000, 5000 y 8000 píxeles. En la segunda fila se muestra la selección con el método de dispersión espacial para (de izquierda a derecha): 797, 1009, 3070, 5069 y 7678 píxeles. En la tercera fila se muestra la selección con el método del cierre convexo para (de izquierda a derecha): 644 (25 cierres convexos), 1462 (50 cierres convexos), 3056 (100 cierres convexos), 5235 (175 cierres convexos) y 7974 píxeles (275 cierres convexos). En la última fila aparece la selección de cierres convexos aleatoriamente (de izquierda a derecha): 725 (25 cierres convexos), 1476 (50 cierres convexos), 2974 (100 cierres convexos), 5197 (175 cierres convexos) y 7232 píxeles (247 cierres convexos).

Dado que para esta secuencia no disponemos de los extrínsecos de la cámara, como en el caso de las secuencias calibradas, vamos a comparar los resultados del seguimiento utilizando un número reducido de puntos con el resultado producido empleando todos los puntos de la plantilla. En la figura 5.22 (derecha), podemos observar la distancia media entre las esquinas estimadas en el seguimiento empleando todos los píxeles de la plantilla y empleando un número de píxeles seleccionado con las diferentes heurísticas propuestas. En este caso el procedimiento de los cierres

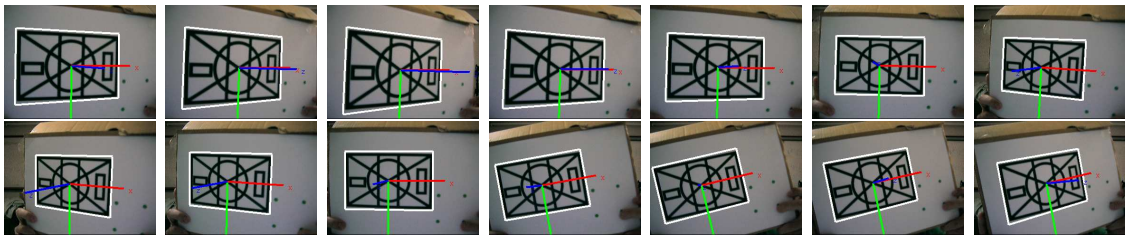


Figura 5.21: Resultados del seguimiento con todos los puntos de la plantilla en el tercer experimento de selección de píxeles. De izquierda a derecha y de arriba a abajo, se muestran una de cada 25 imágenes (empezando por la número 50) de una secuencia de 400. Sobre cada una de ellas aparece enmarcada en blanco la posición estimada de la región objetivo. El sistema de referencia del plano se dibuja en rojo (eje X), verde (eje Y) y azul (eje Z perpendicular al plano).

convexos aleatorios es el peor de los cuatro (error de 0,9 píxeles) mientras que el de las normas máximas aleatorias es el mejor con un error siempre inferior a 0,2 píxeles y por debajo de 0,1 píxeles cuando se eligen más de 2000 píxeles. El método basado en la distribución espacial nos permite obtener un error inferior a 0,2 píxeles pero el error disminuye muy lentamente cuando aumenta el valor de m.

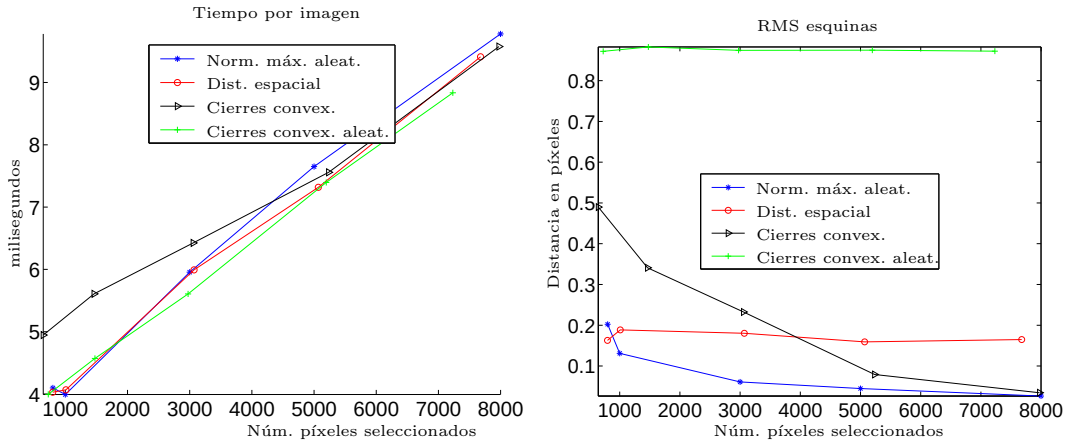


Figura 5.22: Resultados del tercer experimento de selección de píxeles. De izquierda a derecha, gráfica de tiempos, residuo y error en las posición estimada de las cuatro esquinas de la región de interés sobre la imagen, empleando diferente número de píxeles para el seguimiento. En todos los casos se realizaron un máximo de 10 iteraciones del algoritmo en el un único nivel de resolución utilizado.

Con el último experimento pretendemos establecer el comportamiento de los métodos de selección de píxeles con el rostro humano. La plantilla de seguimiento (se puede ver en la figura 5.25) con unas dimensiones de 85×115 píxeles. En este caso también podemos proyectar en dos dimensiones los puntos de la nube Jacobiana para los modelos de movimiento rotación-traslación-escala y proyectivo (ver figura 5.23) dado que los dos mayores autovalores representan la mayor parte de la

variabilidad. Por otra parte en la figura 5.24 tenemos la nube Jacobiana para el caso proyectivo junto con los puntos que elegiríamos con los dos métodos basados en el cierre convexo.

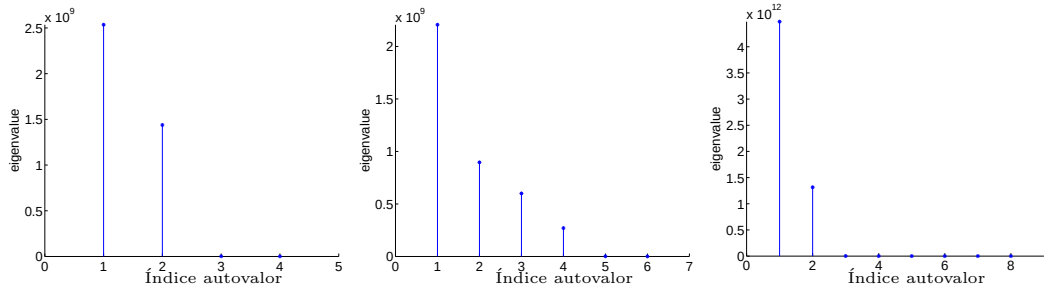


Figura 5.23: Autovalores de la nube jacobiana de la plantilla de seguimiento usada en el último experimento de selección de píxeles. De izquierda a derecha resultados para diferentes modelos de movimiento: rotación-traslación-escala, afín y proyectivo.

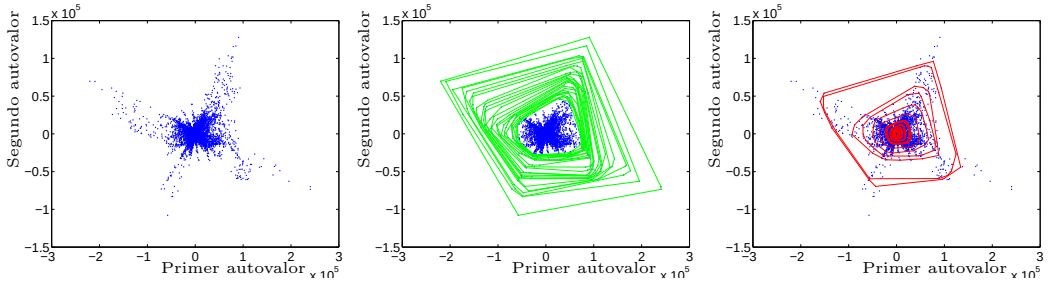


Figura 5.24: Resultados del cierre convexo proyectivo de la plantilla de seguimiento empleada en el último experimento de selección de píxeles. A la izquierda aparece el resultado de proyectar la nube jacobiana del modelo de movimiento proyectivo en dos dimensiones. En el centro los 25 cierres convexos más alejados del centro de la nube. A la derecha aparecen 25 cierres convexos elegidos aleatoriamente de entre el 70 % de los más alejados del centro de la nube jacobiana.

Una vez más, la distribución menos uniforme sobre la plantilla de los píxeles seleccionados (ver figura 5.25), se produce con el método de los cierres convexos. Los puntos elegidos se agrupan alrededor de los bordes con píxeles con mayor $||\mathbf{I}_{\bar{\mu}}(\bar{x})||$, lo que se corresponde con los más exteriores de la plantilla de seguimiento (ver la elección de 25 cierres convexos en la figura 5.25).

Tampoco disponemos de todos los extrínsecos de la cámara, para esta secuencia, así que de nuevo vamos a comparar los resultados del seguimiento utilizando un número reducido de puntos, con el resultado producido empleando todos los puntos de la plantilla. Así que en la figura 5.26 (columna derecha) podemos observar la distancia media entre las esquinas estimadas por el seguimiento con todos los píxeles de la plantilla y las esquinas estimadas con las diferentes heurísticas de selección de píxeles. Atendiendo a la distancia media de las esquinas, una vez más la peor de

Figura 5.25: Píxeles empleados en el último experimento de selección marcados en rojo para el modelo de movimiento proyectivo. En la primera fila se muestra la selección con el método de las normas máximas aleatorias para (de izquierda a derecha): 800, 1000, 3000, 5000 y 7360 píxeles. En la segunda fila se muestra la selección con el método de dispersión espacial para (de izquierda a derecha): 811, 1023, 3053, 5042 y 7419 píxeles. En la tercera fila se muestra la selección con el método del cierre convexo para (de izquierda a derecha): 2776 (105 cierres convexos), 3455 (125 cierres convexos), 5186 (175 cierres convexos), 6977 (230 cierres convexos) y 8141 píxeles (275 cierres convexos). En la última fila aparece la selección de cierres convexos aleatoriamente (de izquierda a derecha): 761 (25 cierres convexos), 2303 (75 cierres convexos), 3715 (125 cierres convexos), 5297 (175 cierres convexos) y 6923 píxeles (228 cierres convexos).

las heurísticas es la de los cierres convexos con la que sólo es posible obtener un error similar a los otros tres métodos de selección a partir de 7000 píxeles (un 70 % del número total de píxeles de la plantilla). Sin embargo, el mejor procedimiento de selección para un número de píxeles por debajo de 5000 (un 50 % de los píxeles de la plantilla) es el de los cierres convexos aleatorios aunque, como en el caso de la plantilla de calibración, aumenta el error con un valor de m cercano a los 7000 píxeles. Por otro lado, con las normas máximas aleatorias obtenemos un comportamiento mucho más regular con una disminución del error de 0,7 a 0,3 píxeles según aumenta el valor de m. En este caso el tiempo en milisegundos empleado en cada imagen varía de acuerdo con el número de iteraciones que sean necesarias para llegar al óptimo (hacemos un máximo de 15 por nivel y utilizamos 2 niveles de resolución) siendo el más lineal (con el número de píxeles elegido) el que corresponde con el método de las normas máximas aleatorias.

La conclusión fundamental de los experimentos es que el método de las normas máximas aleatorias es el que posee un comportamiento más regular. Esto unido a la facilidad de implementación lo convierten en nuestra elección como algoritmo de selección de píxeles.

5.6. Conclusiones

En este capítulo hemos extendido el algoritmo de factorización de Hager y Belhumeur al caso proyectivo (a pesar de que Baker y Matthews [3, 4] aseguran que esto no es posible). Con este algoritmo podemos localizar la cara del usuario en la imagen a pesar de que se encuentre cerca de la cámara. El algoritmo de seguimiento se puede acelerar reduciendo el número de píxeles con una pérdida mínima en precisión y empleando hasta un 57 % menos de tiempo en cada imagen.

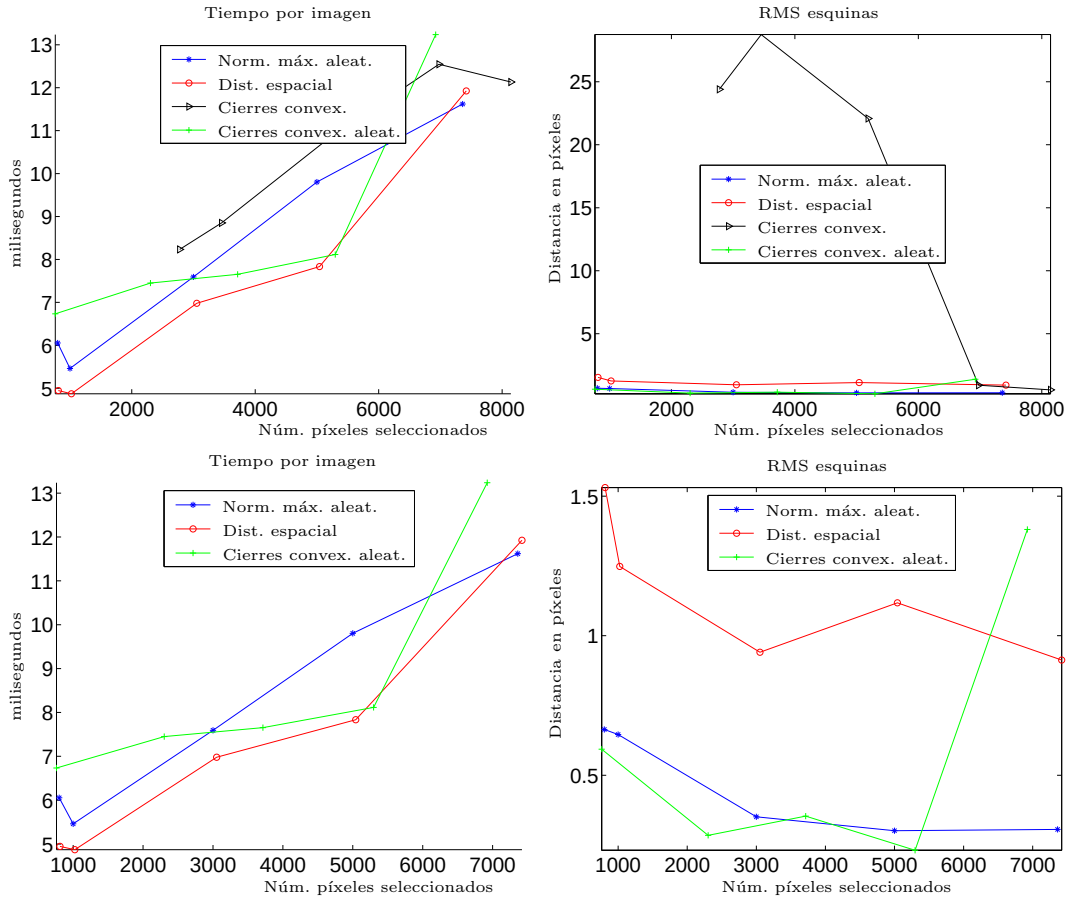


Figura 5.26: Resultados del último experimento de selección de píxeles. En la primera fila, de izquierda a derecha, gráfica de tiempos y error en la posición estimada de las cuatro esquinas de la región de interés sobre la imagen con diferente número de píxeles para el seguimiento. En la segunda fila aparecen los mismos resultados pero sin los resultados del cierre convexo. En todos los casos se realizaron un máximo de 3 iteraciones del algoritmo en cada uno de los dos niveles de resolución utilizados en el seguimiento.

Capítulo 6

Seguidor de la apariencia

Una vez tenemos localizada la cara en su conjunto gracias al seguidor basado en plantillas comenzaremos el análisis de expresiones faciales. A tal fin pretendemos desarrollar un seguidor que no se vea afectado por los cambios de apariencia y que nos permita trabajar en tiempo real.

El número de trabajos que abordan el seguimiento de objetos con modelos lineales de apariencia ha crecido mucho en los últimos años. Una de las limitaciones de estos seguidores es la eficiencia computacional. Una de las primeras aproximaciones al problema de la eficiencia es el trabajo de Hager y Belhumeur [40], en el que se plantea un algoritmo de factorización del Jacobiano eficiente pero sólo válido en el caso de cambios de apariencia debidos a la iluminación. Por su parte Black y Jepson [10], plantean un algoritmo para el caso del cambio de apariencia general que es equivalente en terminos de eficiencia al algoritmo de Lucas y Kanade. Una aproximación eficiente, debida a Baker, está basada en el algoritmo composicional inverso y permite el seguimiento en tiempo real [2].

En resumen, para el caso del seguimiento rígido, en la literatura han aparecido dos algoritmos eficientes de alineación incremental de imágenes: el de la factorización del Jacobiano y el composicional inverso (ver capítulo 4). Sin embargo, para la alineación incremental de imágenes con modelos lineales de apariencia, únicamente ha aparecido una aproximación eficiente basada en el composicional inverso [2]. Como vimos en el capítulo 4, el algoritmo composicional inverso necesita que la función de movimiento, f, forme un semigrupo, lo que constituye una restricción muy fuerte sobre los modelos de movimiento a los que se puede aplicar el algoritmo propuesto por Baker [2]. Dado que la factorización del Jacobiano exige a la f unas condiciones diferentes, es importante desarrollar dentro de este marco de trabajo un algoritmo de alineación incremental de imágenes con modelos lineales de apariencia. Además, ésto nos permitiría emplear la factorización del Jacobiano o el composicional inverso, dependiendo del modelo de movimiento involucrado. En consecuencia, en este capítulo vamos a desarrollar un algoritmo de seguimiento basado modelos lineales de cambio de la apariencia y en la factorización del Jacobiano.

El algoritmo se puede ver como una extensión del introducido por Hager y Belhumeur [40] en la que no imponemos ninguna restricción en el modelo basado en PCA

empleado. También se encuentra relacionado con el trabajo de Black y Jepson [10] pero, en lugar de calcular los parámetros de movimiento mediante un procedimiento de descenso del gradiente en el que el Jacobiano con respecto a los parámetros de movimiento se tiene que recalcular, utilizamos un conjunto de plantillas de movimiento precalculadas que reducen el cómputo necesario durante el seguimiento.

6.1. *Eigentracking* factorizado

Nuestro objetivo es encontrar un algoritmo aditivo para minimizar eficientemente la función de coste de la ecuación (4.49). Una vez más vamos a realizar una expansión en serie de Taylor de $\mathbf{I}(f(\bar{x},\bar{\mu}_t+\delta\bar{\mu}),t+\delta t)$ pero a diferencia del *Eigentracking* original (ver sección 4.2.2) lo haremos alrededor del punto $(\bar{\mu}_t,t)$ (también expandimos alrededor de la coordenada temporal) con lo que se obtendremos

$$\min_{\bar{\mu}} E(\bar{\mu},\bar{c}) = \min_{\bar{\mu}} ||\mathbf{M}_{et}(\bar{\mu}_t,t)\delta\bar{\mu} + \mathbf{I}(f(\bar{x},\bar{\mu}_t),t+\delta t) - \mathbf{B}\bar{c}(t+\delta t)||^2, \tag{6.1}$$

siendo la diferencia fundamental con (4.50) el instante de tiempo en que se calcula $\mathbf{M}_{et}$. En este último caso, al haber desarrollado en serie de Taylor también la coordenada temporal, $\mathbf{M}_{et}$ se instancia en el instante de tiempo t.

Uno de los obstáculos para minimizar (6.1) durante el seguimiento, es el coste computacional de estimar $\mathbf{M}_{et}$ para cada imagen de la secuencia. En esta sección mostraremos cómo se puede factorizar $\mathbf{M}_{et}$ en el producto de dos matrices, $\mathbf{M}_0\mathbf{\Sigma}(\bar{\mu},\bar{c})$, donde $\mathbf{M}_0$ es una matriz constante, que se puede calcular antes del seguimiento.

Tomando derivadas con respecto a $\bar{x}$ en ambos lados de (4.48), la ecuación de constancia de los niveles de gris en el subespacio, obtenemos

$$\left[\left.\frac{\partial \mathbf{I}(\bar{y},t)}{\partial \bar{y}}\right|_{\bar{y}=f(\bar{x},\bar{\mu}_t)}\right]^\top \left[\frac{\partial f(\bar{x},\bar{\mu})}{\partial \bar{x}}\right] = \left[\frac{\partial[\mathbf{B}\bar{c}(t)](\bar{x})}{\partial \bar{x}}\right]. \tag{6.2}$$

Finalmente, de (4.51) y (6.2) obtenemos una nueva expresión para $\mathbf{M}_{et}$,

$$\mathbf{M}_{et}(\bar{\mu}_t,\bar{c}(t)) = \begin{pmatrix} \left(\sum_j \bar{c}(t,j)\left[\frac{\partial[\bar{b}_j](\bar{x}_1)}{\partial\bar{x}}\right]\right)^\top \left[\left.\frac{\partial f(\bar{x},\bar{\mu})}{\partial\bar{x}}\right|_{\bar{x}=\bar{x}_1}\right]^{-1} \left[\left.\frac{\partial f(\bar{x}_1,\bar{\mu})}{\partial\bar{\mu}}\right|_{\bar{\mu}=\bar{\mu}_t}\right] \\ \vdots \\ \left(\sum_j \bar{c}(t,j)\left[\frac{\partial[\bar{b}_j](\bar{x}_N)}{\partial\bar{x}}\right]\right)^\top \left[\left.\frac{\partial f(\bar{x},\bar{\mu})}{\partial\bar{x}}\right|_{\bar{x}=\bar{x}_N}\right]^{-1} \left[\left.\frac{\partial f(\bar{x}_N,\bar{\mu})}{\partial\bar{\mu}}\right|_{\bar{\mu}=\bar{\mu}_t}\right] \end{pmatrix}, \tag{6.3}$$

donde $\bar{b}_j$ es la columna j-ésima de $\mathbf{B}$ y $\bar{c}(t,j)$ es el j-ésimo elemento del vector de apariencia $\bar{c}(t)$.

Sean las matrices

$$\mathbf{B}_\nabla(\bar{x}_i) = \left(\begin{pmatrix} \nabla_u[\bar{b}_1](\bar{x}_i) \\ \vdots \\ \nabla_u[\bar{b}_l](\bar{x}_i) \end{pmatrix}^\top, \begin{pmatrix} \nabla_v[\bar{b}_1](\bar{x}_i) \\ \vdots \\ \nabla_v[\bar{b}_l](\bar{x}_i) \end{pmatrix}^\top \right) \tag{6.4}$$

y

$$\mathbf{C}(t) = \begin{pmatrix} \bar{c}(t,1) & \cdots & \bar{c}(t,l) & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \bar{c}(t,1) & \cdots & \bar{c}(t,l) \end{pmatrix}^{\top}, \tag{6.5}$$

donde u y v son, respectivamente, las coordenadas horizontales y verticales en la imagen. Entonces (6.3) se puede reescribir finalmente como

$$\mathbf{M}_{et}(\bar{\mu}_t, \bar{c}(t)) = \begin{pmatrix} \mathbf{B}_{\nabla}(\bar{x}_1)\mathbf{C}(t) \left[\frac{\partial f(\bar{x},\bar{\mu})}{\partial \bar{x}}\Big|_{\bar{x}=\bar{x}_1} \right]^{-1} \left[\frac{\partial f(\bar{x}_1,\bar{\mu})}{\partial \bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t} \right] \\ \vdots \\ \mathbf{B}_{\nabla}(\bar{x}_N)\mathbf{C}(t) \left[\frac{\partial f(\bar{x},\bar{\mu})}{\partial \bar{x}}\Big|_{\bar{x}=\bar{x}_N} \right]^{-1} \left[\frac{\partial f(\bar{x}_N,\bar{\mu})}{\partial \bar{\mu}}\Big|_{\bar{\mu}=\bar{\mu}_t} \right] \end{pmatrix}. \tag{6.6}$$

Luego $\mathbf{M}_{et}$ se puede expresar en función del gradiente de los vectores de la base del subespacio, $\mathbf{B}_{\nabla}$, que son constantes, y de los parámetros de movimiento y apariencia $(\bar{\mu}, \bar{c})$, que varían con el tiempo. Si elegimos un modelo de movimiento f de tal forma que $\mathbf{C}(t)\left[\frac{\partial f(\bar{x},\bar{\mu})}{\partial \bar{x}}\right]^{-1}\left[\frac{\partial f(\bar{x},\bar{\mu})}{\partial \bar{\mu}}\right] = \mathbf{\Gamma}(\bar{x}_i)\mathbf{\Sigma}(\bar{\mu}, \bar{c})$, entonces $\mathbf{M}_{et}$ se puede factorizar de la siguiente forma

$$\mathbf{M}_{et}(\bar{\mu}_t, \bar{c}(t)) = \begin{pmatrix} \mathbf{B}_{\nabla}(\bar{x}_1)\mathbf{\Gamma}(\bar{x}_1) \\ \vdots \\ \mathbf{B}_{\nabla}(\bar{x}_N)\mathbf{\Gamma}(\bar{x}_N) \end{pmatrix} \mathbf{\Sigma}(\bar{\mu}_t, \bar{c}(t)) = \mathbf{M}_0\mathbf{\Sigma}(\bar{\mu}_t, \bar{c}(t)), \tag{6.7}$$

donde $\mathbf{M}_0$ es una matriz constante y $\mathbf{\Sigma}$ depende de $\bar{c}$ y $\bar{\mu}$. Las columnas de $\mathbf{M}_0$ son las plantillas de movimiento de nuestro algoritmo de seguimiento (ver figura 6.1). Es interesante reflexionar sobre la estructura de $\mathbf{M}_0$. En el caso de la factorización del Jacobiano Hager y Belhumeur [40], si utilizamos un modelo de movimiento con n parámetros, $\bar{\mu} = (\mu_1, \ldots, \bar{\mu}_n)^{\top}$, tendremos que:

$$\mathbf{M}_0 = (\bar{m}_1, \bar{m}_2, \ldots, \bar{m}_n) \tag{6.8}$$

donde $\bar{m}_i$ es el vector de N elementos (el número de píxeles de la plantilla de seguimiento) que contiene la plantilla de movimiento para el parámetro i. Dado que en las plantillas de movimiento tenemos tantos elementos como píxeles, podemos construir una imagen que las represente (ver figura 6.1).

En el caso del *Eigentracking* factorizado la estructura de $\mathbf{M}_0$ difiere ligeramente del anterior, aunque sigue siendo muy parecida. Si utilizamos un modelo de movimiento de n parámetros y una base del subespacio de apariencia con l vectores tendremos que:

$$\mathbf{M}_0 = (\bar{m}_1^1, \ldots, \bar{m}_1^l, \bar{m}_2^1, \ldots, \bar{m}_2^l, \ldots, \bar{m}_n^1, \ldots, \bar{m}_n^l) \tag{6.9}$$

donde $\bar{m}_j^i$ es la plantilla de movimiento correspondiente al j-ésimo parámetro de movimiento para el i-ésimo vector de la base del subespacio de apariencia. Al igual que en el caso del algoritmo de Hager y Belhumeur podemos representar las plantillas de movimiento como imágenes (ver figura 6.1).

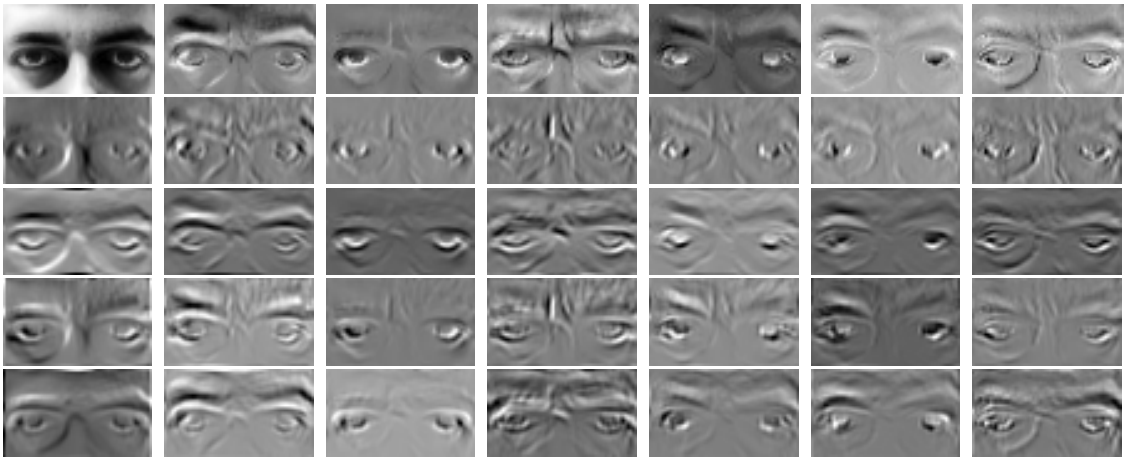


Figura 6.1: Plantillas de movimiento del *Eigentracking* factorizado para la región de los ojos con N=2100 píxeles, l=35 vectores, y n=4 (rotación, traslación y escala). En la primera fila y de izquierda a derecha, la imagen media y los seis primeros vectores de la base de PCA. De la segunda a la cuarta fila, de arriba a abajo, se muestran las plantillas de movimiento correspondientes para la traslación horizontal, vertical, la rotación en el plano y la escala respectivamente.

6.1.1. Minimización de $E(\bar{\mu}, \bar{c})$.

Ya que $\mathbf{M}_{et}$ depende de $\bar{\mu}$ y $\bar{c}$, (4.50) define una función de coste no lineal sobre $\delta\bar{\mu}$ y $\bar{c}$. El algoritmo de optimización que emplearemos primero asume $\bar{c}$ constante y calcula el mínimo de $E(\bar{\mu}, \bar{c})$ con respecto a $\bar{\mu}$,

$$\delta\bar{\mu} = (\mathbf{\Sigma}^\top \mathcal{M} \mathbf{\Sigma})^{-1} \mathbf{\Sigma}^\top \mathbf{M}_0^\top [\mathbf{B}\bar{c}(t+\delta t) - \mathbf{I}(f(\bar{x}, \bar{\mu}_t), \bar{\mu}_t + \delta t)], \tag{6.10}$$

donde $\mathcal{M} = \mathbf{M}_0^\top \mathbf{M}_0$. A continuación minimiza E con respecto a $\bar{c}$ asumiendo $\bar{\mu}$ constante,

$$\bar{c}(t+\delta t) = \mathbf{B}^\top [\mathbf{M}_{et}\delta\bar{\mu} + \mathbf{I}(f(\bar{x}, \bar{\mu}), t+\delta t)]. \tag{6.11}$$

Al igual que ocurría en el caso del *Eigentracking* original, dado que el cálculo de $\bar{\mu}$ y de $\bar{c}$ dependen el uno del otro vamos a asumir que la apariencia del objeto en seguimiento varía lentamente. Esto nos permite emplear $\bar{c}(t)$ como punto de partida para el cálculo de $\delta\bar{\mu}$ en el instante $t + \delta t$. Por otro lado, es razonable asumir que ambos espacios de minimización (rígido y no rígido) son casi independientes pues la base del subespacio de deformaciones se calcula después de registrar las imágenes y eliminar la componente rígida del movimiento, lo cual equivaldría a una proyección en el espacio de deformaciones.

El término $\mathbf{M}_{et}\delta\bar{\mu}$ en (6.11) es la variación de los niveles de gris en $\mathbf{I}$ debido a un movimiento de magnitud $\delta\bar{\mu}$. Intuitivamente, la ecuación (6.11) establece que los parámetros de apariencia se calculan mediante la proyección sobre el subespacio de la imagen rectificada corregida para tener en cuenta el movimiento $\delta\bar{\mu}$. Una vez que tenemos $\bar{c}$, podemos mejorar la estimación de $\delta\bar{\mu}$ mediante el uso de (6.10) una

vez más. Normalmente dos o tres iteraciones de este proceso son suficientes para alcanzar una situación estable.

En resumen, los pasos necesarios en el seguimiento se muestran en el algoritmo 6.1.

- **Pre-cálculo:**
  1. Calcular los gradientes de los vectores de la base, $\nabla[b_i](\bar{x})$.
  2. Calcular todas las matrices $\Gamma(\bar{x})$.
  3. Calcular y almacenar $\mathbf{M}_0$.
  4. Calcular y almacenar $\mathcal{M}$.
- **En cada iteración:**
  1. Rectificar $I(\bar{z}, t+\tau)$ para calcular $\mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\tau)$.
  2. Calcular la imagen reconstruida, $\mathbf{B}\bar{c}(t)$.
  3. Calcular $\mathcal{E} = [\mathbf{B}\bar{c}(t) - \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t)]$.
  4. Calcular $\mathbf{\Sigma}$.
  5. Calcular $\mathbf{\Sigma}^\top \mathbf{M}_0^\top$.
  6. Calcular $\mathbf{\Sigma}^\top \mathbf{M}_0^\top \mathcal{E}$.
  7. Calcular $(\mathbf{\Sigma}^\top \mathcal{M} \mathbf{\Sigma})^{-1}$.
  8. Empleando (6.10) calcular $\delta\bar{\mu}$.
  9. Usando (6.11) calcular $\bar{c}(t+\delta t)$ a partir de $\delta\bar{\mu}_{t+\tau}$.

Algoritmo 6.1: *Eigentracking* factorizado

Sea l el número de vectores de la base, n el número de parámetros de movimiento y N el número de píxeles de la región a seguir. Entonces el coste computacional de la parte de pre-cálculo del algoritmo se muestra en el cuadro 6.1.

Paso (1)	Paso (2)	Paso (3)	Paso (4)	Total
$\mathrm{O}(lN)$	$\mathrm{O}(lN)$	$\mathrm{O}(l^2nN)$	$\mathrm{O}(l^2n^2N)$	$\mathrm{O}(l^2n^2N)$

Cuadro 6.1: Orden de complejidad en número de operaciones de la parte de pre-cálculo del algoritmo de *Eigentracking* factorizado.

El orden de complejidad necesario para hacer una iteración del algoritmo se muestra en el cuadro 6.2. El orden de complejidad final viene marcado por los pasos (5), $\mathrm{O}(kn^2N)$, y (7), $\mathrm{O}(k^2n^3)$. Sólo cuando el número de píxeles, N, es muy pequeño (típicamente imágenes 10×10) el paso (7) domina en el orden de complejidad. Cuando se emplean imágenes de tamaño 20×20 o más grandes, la mayor parte del tiempo se emplea en multiplicar y trasponer la matriz Jacobiana, $(\mathbf{\Sigma}\mathbf{M}_0)^\top$. Mediante la

optimización del procedimiento de multiplicación de matrices podremos mejorar el rendimiento en este paso del algoritmo.

Paso (1)	Paso (2)	Paso (3)	Paso (4)	Paso (5)	Paso (6)
$O(nN)$	$O(lN)$	$O(N)$	$O(l)$	$O(ln^2N)$	$O(nN)$

Paso (7)	Paso (8)	Paso (9)	Total
$O(l^2n^3)$	$O(n^2)$	$O(lN+nN)$	$O(ln^2N+l^2n^3)$

Cuadro 6.2: Orden de complejidad en número de operaciones de una iteración del algoritmo de *Eigentracking* factorizado.

6.1.2. Algunos modelos de movimiento usuales

En esta sección presentaremos cómo emplear el algoritmo de seguimiento previo con algunos modelos de movimiento usados habitualmente en Visión por Computador.

Modelo de rotación, traslación y escala

Este modelo de movimiento se puede describir mediante cuatro parámetros, $\bar{\mu} = (t_u, t_v, \theta, s)$, correspondientes a rotación, traslación y escala, $f(\bar{x}, \bar{\mu}) = s\mathbf{R}(\theta)\bar{x} + \bar{t}$, donde $\bar{x} = (u, v)^\top$, $\bar{t} = (t_u, t_v)^\top$ y $\mathbf{R}(\theta)$ es una matriz de rotación 2D. Tomando derivadas de f con respecto a $\bar{x}$ y $\bar{\mu}$,

$$\left[\frac{\partial f(\bar{x},\bar{\mu})}{\partial \bar{x}}\right] = s\mathbf{R}(\theta), \tag{6.12}$$

$$\left[\frac{\partial f(\bar{x},\bar{\mu})}{\partial \bar{\mu}}\right] = \left(\mathbf{I}_{2\times 2} \,|\, -s\mathbf{R}(\theta)\begin{pmatrix} -v \\ u \end{pmatrix} \,|\, \mathbf{R}(\theta)\begin{pmatrix} u \\ v \end{pmatrix}\right), \tag{6.13}$$

donde $\mathbf{I}_{d\times d}$ es la matriz identidad $d \times d$. Introduciendo (6.12) y (6.13) en la ecuación (6.6), obtenemos la factorización:

$$\mathbf{\Gamma}(\bar{x}_i) = \left(\mathbf{I}_{2l\times 2l},\ \begin{pmatrix} -v_i\mathbf{I}_{l\times l} & u_i\mathbf{I}_{l\times l} \\ u_i\mathbf{I}_{l\times l} & v_i\mathbf{I}_{l\times l}\end{pmatrix}\right),$$

$$\mathbf{\Sigma}(\bar{\mu},\bar{c}) = \begin{pmatrix} \mathbf{C}\frac{1}{s}\mathbf{R}(-\theta) & 0 \\ 0 & \mathbf{C}\begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{s}\end{pmatrix}\end{pmatrix}.$$

Para este modelo $\mathbf{M}_0$ tiene dimensiones $N \times 4l$ y las dimensiones de $\mathbf{\Sigma}$ son $4l \times 4$.

Modelo afín

El modelo de movimiento afín en 2D se puede escribir como

$$f(\bar{x},\bar{\mu}) = \underbrace{\begin{pmatrix} a & c \\ b & d\end{pmatrix}}_{\mathbf{A}}\bar{x} + \begin{pmatrix} e \\ f\end{pmatrix},$$

donde $\mathbf{A}$ es una matriz no singular y $\bar{\mu} = (e, f, a, b, c, d)^\top$ son los seis parámetros del modelo. Tomando derivadas de f con respecto a $\bar{x}$ y $\bar{\mu}$,

$$\begin{aligned} \left[\frac{\partial f(\bar{x}, \bar{\mu})}{\partial \bar{x}}\right] &= \mathbf{A}, \\ \left[\frac{\partial f(\bar{x}, \bar{\mu})}{\partial \bar{\mu}}\right] &= (\mathbf{I}_{2\times 2} \mid u\mathbf{I}_{2\times 2} \mid v\mathbf{I}_{2\times 2})\,. \end{aligned} \tag{6.14}$$

A partir de (6.14) y (6.6), se llega a la factorización buscada:

$$\boldsymbol{\Gamma}(\bar{x}_i) = (\mathbf{I}_{2l\times 2l} | u_i\mathbf{I}_{2l\times 2l} | v_i\mathbf{I}_{2l\times 2l})\,, \tag{6.15}$$

$$\boldsymbol{\Sigma}(\bar{\mu}, \bar{c}) = \begin{pmatrix} \mathbf{C}\mathbf{A}^{-1} & 0 & 0 \\ 0 & \mathbf{C}\mathbf{A}^{-1} & 0 \\ 0 & 0 & \mathbf{C}\mathbf{A}^{-1} \end{pmatrix}. \tag{6.16}$$

En este caso $\mathbf{M}_0$ resulta de dimensiones $N \times 6l$ y $\boldsymbol{\Sigma}$ de $6l \times 6$.

Modelo proyectivo

Sean $\bar{x} = (u, v)^\top$ y $\bar{x}_h = (r, s, \lambda)^\top$, respectivamente, las coordenadas cartesianas y proyectivas de un píxel de la imagen. Las dos se encuentran relacionadas mediante: $\bar{x}_h = (r, s, \lambda)^\top \rightarrow \bar{x} = w(\bar{x}_h) = (r/\lambda, s/\lambda)^\top = (u, v)^\top;\ \lambda \neq 0$. La transformación lineal proyectiva 2D, se puede escribir como sigue

$$f(\bar{x}_h, \bar{\mu}) = \mathbf{H}\bar{x}_h = \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & 1 \end{pmatrix} \begin{pmatrix} r \\ s \\ \lambda \end{pmatrix},$$

donde $\bar{\mu} = (a, b, c, d, e, f, g, h)^\top$. Ahora $\mathbf{B}_\nabla(\bar{x}_i)$ posee un conjunto extra de columnas asociado a los gradientes de la coordenada homogénea [1],

$$\mathbf{B}^P_\nabla(\bar{x}_i) = \left(\begin{pmatrix} \nabla_r[\bar{b}_1](\bar{x}_i) \\ \vdots \\ \nabla_r[\bar{b}_k](\bar{x}_i) \end{pmatrix}^\top, \begin{pmatrix} \nabla_s[\bar{b}_1](\bar{x}_i) \\ \vdots \\ \nabla_s[\bar{b}_k](\bar{x}_i) \end{pmatrix}^\top, \begin{pmatrix} \nabla_\lambda[\bar{b}_1](\bar{x}_i) \\ \vdots \\ \nabla_\lambda[\bar{b}_k](\bar{x}_i) \end{pmatrix}^\top \right) \tag{6.17}$$

y la matriz $\mathbf{C}$ ahora es

$$\mathbf{C}^P(t) = \begin{pmatrix} \bar{c}(t,1) & \cdots & \bar{c}(t,l) & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \bar{c}(t,1) & \cdots & \bar{c}(t,l) & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \bar{c}(t,1) & \cdots & \bar{c}(t,l) \end{pmatrix}^\top. \tag{6.18}$$

Tomando derivadas de f con respecto a $\bar{x}_h$ y $\bar{\mu}$,

$$\left[\frac{\partial f(\bar{x}_h, \bar{\mu})}{\partial \bar{x}_h}\right] = \mathbf{H}, \tag{6.19}$$

$$\left[\frac{\partial f(\bar{x}_h, \bar{\mu})}{\partial \bar{\mu}}\right] = (r\mathbf{I}_{3\times 3} \mid s\mathbf{I}_{3\times 3} \mid \lambda\mathbf{I}_{1-2})\,, \tag{6.20}$$

[1] $\nabla_{\bar{x}_h}\mathbf{I}(\bar{x}) = \left[\frac{\partial \mathbf{I}}{\partial u}, \frac{\partial \mathbf{I}}{\partial v}, -r\frac{\partial \mathbf{I}}{\partial u} - s\frac{\partial \mathbf{I}}{\partial v}\right]^\top$, ver sección 5.1

donde $\mathbf{X}_{a-b}$ es la matriz compuesta por las columnas desde a hasta b de la matriz $\mathbf{X}$. Entonces, a partir de (6.17–6.20) y (6.6) se calcula la factorización de $\mathbf{M}_e t$:

$$\mathbf{\Gamma}(\bar{x}_i) = (r_i \mathbf{I}_{3l\times 3l} \mid s_i \mathbf{I}_{3l\times 3l} \mid \lambda_i \mathbf{I}_{3l\times 3l}) \tag{6.21}$$

$$\mathbf{\Sigma}(\bar{\mu}, \bar{c}) = \begin{pmatrix} \mathbf{C}^P\mathbf{H}^{-1} & 0 & 0 \\ 0 & \mathbf{C}^P\mathbf{H}^{-1} & 0 \\ 0 & 0 & \mathbf{C}^P\mathbf{H}^{-1}_{1-2} \end{pmatrix}. \tag{6.22}$$

Ahora las dimensiones de $\mathbf{M}_0$ y $\mathbf{\Sigma}$ son $N \times 9l$ y $9l \times 8$ respectivamente.

6.2. *Eigentracking* modular factorizado

Un modelo modular basado en subespacios es una partición del vector original de datos en subconjuntos (módulos) de forma que se pueda calcular un modelo basado en subespacios para cada uno de los módulos. Esto nos permite construir un modelo más flexible, compacto, preciso y mejor condicionado de las regiones de interés [65]. En nuestro caso emplearemos diferentes subespacios para cada uno de los ojos y para la boca.

Sea $\{\mathbf{B}_1, \cdots, \mathbf{B}_r\}$ el conjunto de bases para los subespacios de todos los módulos. Entonces la matriz $\mathbf{B}_{me}$ para el eigentracking modular se puede escribir como:

$$\mathbf{B}_{me} = \begin{bmatrix} \mathbf{B}_1 & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & \mathbf{B}_r \end{bmatrix}, \tag{6.23}$$

que es una matriz diagonal por bloques que representa el conjunto de regiones que componen la imagen. La apariencia de cada región viene modelada por la base del subespacio $\mathbf{B}_i$. El vector de parámetros de apariencia será $\bar{c} = (\bar{c}_1^\top, \cdots, \bar{c}_r^\top)^\top$, donde $\bar{c}_i$ es el vector de parámetros del módulo i. En el cálculo de $\mathbf{M}_0$, los gradientes de $\mathbf{B}_{me}$ se obtienen independientemente para cada $\mathbf{B}_i$ y, como se vio en la sección anterior, se introducen en $\mathbf{B}_\nabla$. Finalmente, $g_i(\bar{\mu})$ es una función que relaciona los parámetros de movimiento del módulo i con un sistema de referencia común a todos. Asumimos que todos los módulos pertenecen al mismo objeto 3D y poseen el mismo movimiento rígido. Los pasos necesarios en el algoritmo de *Eigentracking* modular factorizado se muestran en el algoritmo 6.2.

6.3. Experimentos

Hemos implementado nuestro algoritmo en C++ en el sistema operativo GNU/Linux. Se emplean las rutinas de rectificado de imágenes de la biblioteca de programación Intel IPL (Image Processing Library) y la rutina *dgemm* para la multiplicación de matrices de BLAS (Basic Linear Algebra Soubrutines), en una versión optimizada para el pentium IV (de la implementación que aparece en la biblioteca de programación ATLAS). No se ha realizado ninguna otra optimización especial en el código

- **Pre-cálculo:**
  1. Calcular y almacenar $\mathbf{M}_0$ empleando B_{me}.
  2. Calcular y almacenar $\mathcal{M}$.
- **En cada iteración:**
  1. Para cada región i hacer:
     a) Rectificar $\mathbf{I}(z, t+\tau)$ para calcular $\mathbf{I}(f(x, g_i(\bar{\mu}_t)), t+\delta t)$.
     b) $\mathcal{E}_i = [\mathbf{B}_i \bar{c}_i(t) - \mathbf{I}(f(\bar{x}, g_i(\bar{\mu}_t)), t+\delta t)]$.
  2. El vector de error es ahora $\mathcal{E} = (\mathcal{E}_1^\top, \cdots, \mathcal{E}_r^\top)^\top$.
  3. Calcular $\mathbf{\Sigma}(\bar{\mu}_t, \bar{c}(t))$.
  4. Usando (6.10) calcular $\delta\bar{\mu}$ empleando el nuevo $\mathcal{E}$.
  5. Usando (6.11) calcular $\bar{c}(t+\tau)$ empleando $\delta\bar{\mu}$ and $\mathbf{B}_{me}$.
  6. Actualizar $\bar{\mu}_{t+\delta t} = \bar{\mu}_t + \delta\bar{\mu}$.
  7. Actualizar cada vector $\bar{c}_i$ con $\bar{c}(t+\delta t)$.

Algoritmo 6.2: *Eigentracking* modular factorizado

Figura 6.2: Algunos ejemplos de la secuencia empleada en el primer experimento.

actual (p.ej. analizar la estructura de la matriz $(\Sigma\mathbf{M}_0)^\top$). El computador en el que realizamos las pruebas posee un procesador Pentium IV a 2.4 GHz con 512 KBytes de caché y 512 MBytes de memoria RAM. Las secuencias de imágenes se capturaron con una cámara IEEE 1394 (*firewire*) Sony VL500 y se almacenaron en disco. En todos los experimentos la construcción de los modelos de PCA necesarios para el seguimiento se realiza automáticamente a partir de secuencias de imágenes de entrenamiento (ver apéndice B).

En el primer experimento comprobaremos el rendimiento del algoritmo en función del tiempo necesario para realizar una iteración con diferentes modelos de movimiento (n), número de píxeles (N), y dimensiones del subespacio (l). En este caso usamos una secuencia con 595 imágenes con ambos ojos y cejas para construir el modelo de PCA (ver figura 6.2). El tiempo necesario por iteración en milisegundos se muestra en el cuadro 6.3.

	N=136 × 56			N=68 × 28		
	l=7	l=13	l=44	l=7	l=13	l=39
Proyectivo (n=8)	29.9	41.1	98	5.6	7.7	16.9
Afín (n=6)	20.3	28.8	71.6	4.1	5.2	11.5
RTE (n=4)	14.3	21.5	57.1	3.3	4.3	8.6

Cuadro 6.3: Tiempo por iteración en milisegundos.

En el cuadro 6.4 se muestra el número de imágenes por segundo procesadas cuando el algoritmo realiza dos iteraciones por imagen. Con el algoritmo propuesto, se pueden procesar imágenes a la frecuencia de la señal de vídeo estándar con cualquier región de 68×28 píxeles cuya apariencia se pueda modelar con un subespacio de dimensiones inferiores a 40. Adicionalmente, dada la estructura especial de los niveles de gris del rostro humano, al presentar principalmente componentes de baja frecuencia, se puede seguir sin problemas con un subespacio de dimensión reducida (por ejemplo, l=7) para el que se pueden alcanzar de 16,7 imágenes por segundo a 151,5 imágenes por segundo, dependiendo del número de píxeles (N) y de la complejidad del modelo de movimiento (n).

	N=136 × 56			N=68 × 28		
	l=7	l=13	l=44	l=7	l=13	l=39
Proyectivo (n=8)	16.7	12.1	5.1	89.3	65	29.6
Afín (n=6)	24.6	17.4	7	122	96.1	43.5
RTE (n=4)	35	23.3	8.8	151.5	116.3	58.1

Cuadro 6.4: Imágenes por segundo con dos iteraciones por imagen.

En el segundo experimento estudiamos el rendimiento del algoritmo propuesto empleando un modelo de subespacios modular y el modelo de movimiento rotación-traslación-escala (en adelante RTE). En este caso se utiliza una secuencia de 595 imágenes ejemplo para cada ojo y 648 imágenes para la boca. La dimensión del subespacio de la boca es 32, mientras que el de cada ojo es únicamente 6. Los ejemplos de ojos y boca son imágenes de 28×23 y 43×30 píxeles, respectivamente. En la figura 6.3 se muestran algunas de las imágenes de la secuencia con la posición estimada de cada región marcada en rojo. Al lado derecho de cada imagen se muestran las imágenes rectificadas (arriba) y reconstruidas (abajo). En nuestra implementación usamos ecualización del histograma de los niveles de gris tanto en el PCA como en el seguimiento, para normalizar las imágenes con respecto a los cambios de iluminación, lo que proporciona cierta robustez frente a los cambios de la geometría del sistema de iluminación. Esta es la causa de algunos errores de reconstrucción. Sin embargo, el procedimiento de minimización es bastante robusto a esos errores y el proceso de seguimiento continua sin problemas. En el procesamiento de esta secuencia realizamos tres iteraciones por imagen para lograr un rendimiento de 20 imágenes por segundo.

En el tercer experimento se muestra el comportamiento para un modelo de movimiento proyectivo. Una vez más, la secuencia de entrenamiento empleada para el PCA (la misma que en el primer experimento) es diferente de la usada en el seguimiento. En este caso la dimensión del subespacio era 13, el tamaño de las imágenes de ejemplo empleadas 68×28 píxeles y se procesaron 32 imágenes por segundo realizando tres iteraciones por imagen. La diferencia con las 65 imágenes por segundo mostradas en el cuadro 6.3 para dos iteraciones, se debe principalmente a la sobrecarga de dibujar los resultados, cargar las imágenes desde disco y realizar una iteración adicional. En el experimento la cabeza realiza rotaciones moderadas fuera

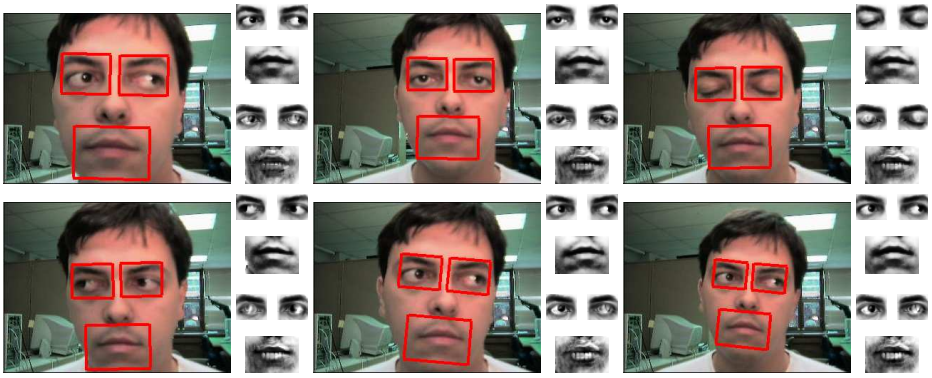


Figura 6.3: Seguimiento de la apariencia con un modelo de movimiento RTE modular. Los resultados son para una secuencia de 1560 imágenes. En rojo aparece resaltada la posición estimada de cada región de la cara (ojos y boca) sobre cada imagen y a su derecha: las imágenes rectificadas de cada módulo (arriba) junto con las imágenes reconstruidas de cada módulo (abajo).

del plano y el seguidor es capaz de no perder el objetivo. En la figura 6.4 se pueden observar los resultados del experimento. La posición estimada de los ojos aparece superpuesta en rojo sobre la imagen actual y a su derecha se muestran las imágenes rectificada (arriba) y reconstruida (abajo).

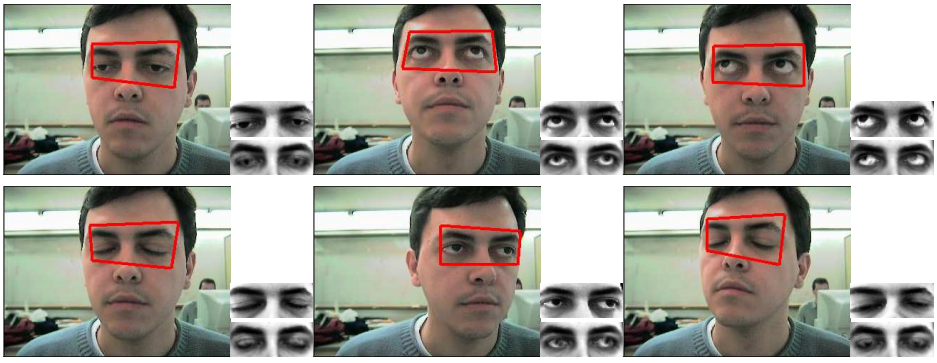


Figura 6.4: Seguimiento proyectivo de la apariencia. Resultados para una secuencia de 643 imágenes. En rojo aparece resaltada la posición estimada de la región de los ojos sobre cada imagen y a su derecha: la imagen rectificada (arriba) junto con la imagen reconstruida (abajo).

En el siguiente experimento probaremos el algoritmo para una situación ideal en la que el modelo de apariencia es el óptimo, esto es, realizaremos el seguimiento sobre la misma secuencia empleada en el entrenamiento del modelo de PCA. Emplearemos un modelo de apariencia modular para la boca y para ambos ojos, un modelo de movimiento proyectivo y haremos dos iteraciones por imagen. Tal y como se puede observar en la figura 6.5, el seguimiento se desarrolla bastante bien y la apariencia se estima correctamente en todas las imágenes de la secuencia. En este experimento

el seguidor es capaz de procesar 18 imágenes por segundo para el modelo proyectivo, 26 para el afín y 34 para el RTE [2].

Figura 6.5: Seguimiento modular de la apariencia con un modelo de movimiento proyectivo. Se muestran los resultados para una secuencia de 798 imágenes. Esta secuencia se usó también en el entrenamiento del modelo de apariencia. En rojo aparece resaltada la posición estimada de cada región de la cara (ojos y boca) sobre cada imagen y a su derecha: las imágenes rectificadas de cada módulo (arriba) junto con las imágenes reconstruidas de cada módulo (abajo).

En el último experimento probaremos el comportamiento del algoritmo en una secuencia más complicada. En esta prueba empleamos una secuencia lo bastante larga para emplear la mitad de la secuencia en el entrenamiento del modelo de apariencia y la otra mitad en el seguimiento. El modelo de apariencia empleado es modular para la boca (35×23 píxeles) y ambos ojos (33×35 píxeles cada uno), un modelo de movimiento RTE y realizamos cuatro iteraciones por imagen en el procedimiento de optimización. Tal y como se muestra en la figura 6.6, el seguimiento se realiza correctamente y los parámetros de apariencia y movimiento se estiman bien en todas las imágenes. En este experimento el seguidor es capaz de procesar 13 imágenes por segundo con el modelo RTE[3].

[2]El número de imágenes por segundo procesadas en este experimento incluyen el tiempo necesario para la lectura de imágenes y la presentación de resultados.

[3]El número de imágenes por segundo procesadas incluye el tiempo necesario para cargar las imágenes de disco y mostrar los resultados

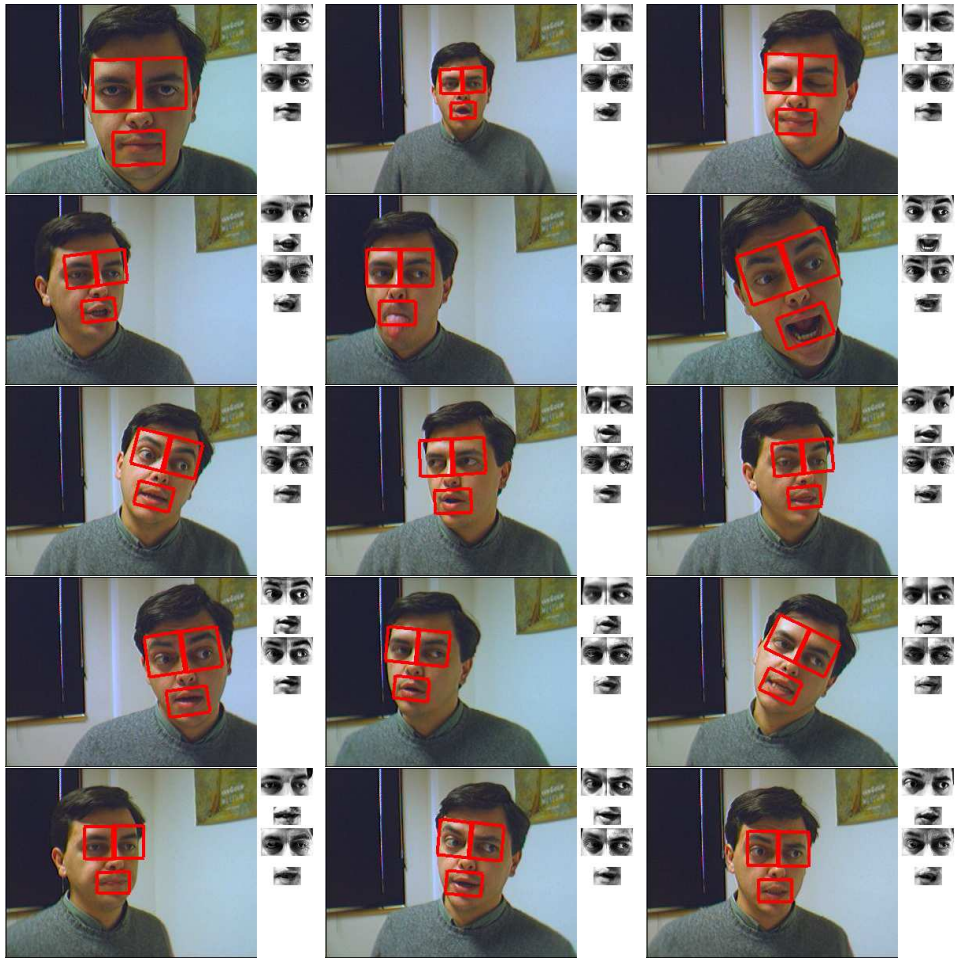


Figura 6.6: Seguimiento de la apariencia con un modelo de movimiento RTE. La secuencia empleada en el experimento tenía 4787 imágenes, la mitad se usó para el entrenamiento del modelo de apariencia (2720 imágenes) y la otra mitad en el seguimiento (2067 imágenes). En rojo aparece resaltada la posición estimada de cada región de la cara (ojos y boca) sobre cada imagen y a su derecha: las imágenes rectificadas de cada módulo (arriba) junto con las imágenes reconstruidas de cada módulo (abajo).

6.4. Conclusiones

La eficiencia es un factor clave para que los procedimientos basados en la apariencia sean útiles en aplicaciones de seguimiento. En este capítulo hemos presentado un procedimiento eficiente para el seguimiento empleando un modelo de subespacios lineales de la apariencia del objetivo. La eficiencia se obtiene precalculando el conjunto de plantillas de movimiento que se derivan de la factorización del Jacobiano de la imagen. También en este capítulo se ha mostrado cómo realizar esta factorización para los modelos de movimiento usuales: RTE, afín y proyectivo. En los trabajos de Baker y Matthews [3, 4] se asegura que la factorización del Jacobiano

no se puede utilizar con modelos de movimiento proyectivos. En este capítulo no solo hemos demostrado que la factorización se puede aplicar al seguimiento con modelos proyectivos, sino que también es aplicable al seguimiento basado en la apariencia con modelos lineales.

En los experimentos realizados hemos mostrado, con una implementación no optimizada, que se puede procesar fácilmente imágenes a la frecuencia de la señal de vídeo en el seguimiento del rostro humano o cualquier otra región de una imagen con tamaño moderado y textura con componentes de baja frecuencia.

En el proceso de minimización se han supuesto aproximadamente independientes (ortogonales) los subespacios asociados al movimiento rígido y no rígido. Esta hipótesis se ha visto confirmada en que normalmente un par de iteraciones son suficientes para que los algoritmos 6.1 y 6.2 converjan.

Existen todavía algunas cuestiones abiertas bastante importantes en las que se continúa trabajando, entre otras, cómo tratar los cambios de iluminación y las oclusiones sin perder la eficiencia computacional.

Parte IV

Aplicación del análisis de expresiones faciales

Capítulo 7

Animación facial

La animación por ordenador de personajes tiene una importancia cada vez mayor en el mundo de los de los vídeo-juegos, la televisión o el cine. Animar de forma realista un personaje creado por ordenador no es nada fácil y en los últimos años se recurre a la captura del movimiento de un actor real. Los sistemas comerciales de captura de movimiento son invasivos en el sentido de que necesitan una iluminación controlada y marcadores (normalmente dispositivos reflectantes o magnéticos) que se colocan sobre la piel del actor o van sobre un traje especial que éste debe vestir [53, 51, 48, 49, 52]. Debido a su propia filosofía de funcionamiento, el uso de estos sistemas es muy incómodo cuando las sesiones de trabajo son muy largas (ver figura7.1).

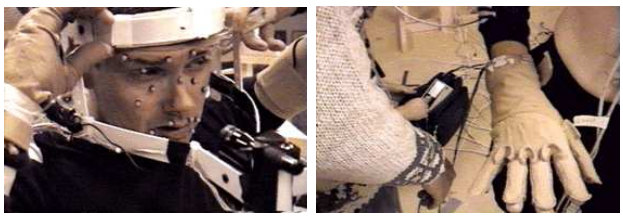


Figura 7.1: Sistemas de captura de movimiento comerciales. El actor debe vestir trajes especiales o colocarse marcadores sobre la piel.

Una de las posibles aplicaciones del análisis facial es la reanimación de personajes virtuales a partir del análisis de una secuencia de imágenes de una persona real. En el trabajo desarrollado en la tesis hasta el momento podemos seguir el rostro a pesar de las deformaciones producidas por el movimiento de cejas, ojos o boca. En este capítulo se propone una aplicación del sistema de seguimiento construido para animar un modelo 3D del rostro humano (ver las características del modelo en el apéndice C) estimando los parámetros de animación a partir del seguidor basado en la apariencia (ver figura 7.2).

Figura 7.2: Reanimación de un modelo 3D del rostro humano, sin marcadores, dirigida por la actuación de una persona

7.1. Introducción

En la literatura han aparecido diferentes formas de animar una cara a partir de datos reales tales como: animación dirigida por la voz [13, 29] y animación dirigida por el movimiento o la actuación de un humano [88, 5]. Las ventajas de la animación dirigida por un flujo de datos real sobre la animación tradicional (manual) son, principalmente: la rapidez con la que se puede generar una nueva animación, y el potencial de producir unos resultados mucho más realistas. Por otro lado, podemos clasificar las diferentes aproximaciones al problema dependiendo de si se trata de animar un modelo 2D [5, 14, 29], 3D [24] o ambos tipos [88, 13].

Los métodos actuales de captura de movimiento emplean marcadores sobre la cara del actor y estiman la posición 3D o 2D de los mismos. Si la forma de la cara en los datos de entrada no se parece lo bastante al modelo a animar, entonces se necesitará un método para adecuar el movimiento capturado al nuevo modelo. Podemos dividir los métodos de adecuación del movimiento en dos categorías: parametrización y modificación del movimiento.

Los métodos de modificación del movimiento facial tratan de adaptar las deformaciones de los vértices producidas por las expresiones faciales al nuevo modelo. Por ejemplo, si la frente del modelo origen es más vertical que la del modelo destino, la elevación de las cejas se debe modificar para que se produzca sobre el nuevo plano de la frente. En el trabajo de Jun-yong y Newmann [61], se plantean una serie de algoritmos y heurísticas para transportar el movimiento de una expresión facial entre dos modelos con estructura superficial (número de vértices y conexiones entre los mismos) y proporciones diferentes. Todo ello a partir de un número reducido de correspondencias entre los dos modelos introducidas manualmente por el usuario.

Los métodos basados en la parametrización tratan de describir el movimiento con un conjunto de valores, que aplicados a cualquier modelo facial, producen una expresión similar. Entre los sistemas parametrizados podemos distinguir los que emplean una codificación de las expresiones faciales *ad-hoc* [13, 29, 14, 88, 5, 24] y los que emplean una codificación estándar como FACS (Facial Action Coding System) [18, 89] o MPEG-4 FAPS (Facial Animation Parameters Set) [96, 97, 37]. Cuanto mayor sea el nivel de los parámetros de animación (esto es, cuanto más alejados del movimiento particular de cada vértice del modelo de animación estén)

más difícil será estimarlos. Fundamentalmente porque la relación entre las medidas sobre la imagen y los parámetros de control será mucho más débil.

En las siguientes secciones repasaremos algunos de los sistemas de animación dirigida por un flujo de datos real aparecidos en la literatura.

7.1.1. Animación dirigida por el movimiento

La animación de personajes basada en el análisis de una secuencia de vídeo ha recibido una atención creciente en el mundo de los gráficos y la Visión por Computador [13, 29, 14, 88, 5, 24].

Uno de los primeros sistemas de animación de un modelo 3D del rostro basado en Visión por Computador se debe a Terzopoulos y Waters [88]. Se pretendía animar un modelo 3D del rostro humano que simulaba el comportamiento físico de la piel mediante la activación de diferentes músculos. En una secuencia de vídeo se seguían las cejas, el mentón, el contorno de la boca y las arrugas nasolabiales mediante contornos activos. La animación del modelo, se realizaba trasladando el movimiento de ciertos puntos, sobre los contornos seguidos, a activaciones de los músculos correspondientes. Una de las limitaciones del sistema es que necesitaba de maquillaje en el actor y una posición frontal para que el seguimiento funcionase.

Bascle [5] utilizó contornos activos para el análisis facial. La principal aportación de su aproximación es la separación entre orientación de la cabeza y la forma que adoptan los contornos activos mediante Descomposición en Valores Singulares (SVD). El resultado es que se puede reanimar la posición que adopta el modelo 2D de contornos en cualquier otra orientación de la cabeza. Una vez más, el principal inconveniente es que se necesita maquillaje en algunos de los contornos para realizar el seguimiento. Con el resultado del algoritmo se podría animar un modelo 3D, como en [88], o 2D mediante técnicas de *morphing*. La diferencia fundamental con el trabajo de Terzopoulos y Waters [88], es que permite recuperar la forma de los contornos activos independientemente de la orientación de la cabeza.

Siguiendo con los sistemas basados en características geométricas 2D, nos encontramos con el trabajo de Ian Buck [16]. Se trata de animar un modelo 2D que consiste en una caricatura de la cara dibujada a mano. Para ello se establecen manualmente cuáles de las imágenes del usuario, de las que se conocen los parámetros de seguimiento, se corresponden con 6 dibujos de la boca y 4 de los ojos con diferentes expresiones. En el seguimiento se emplea el color de los labios y de las pupilas para localizar la región de la boca y el centro de los ojos en cada imagen. A partir de la localización somera de ojos y boca, se calcula mediante métodos *ad-hoc* la posición de las pupilas, la distancia entre ellas, la apertura de los párpados, la posición de las cejas e incluso la apertura de la boca. La imagen de la caricatura se obtiene haciendo morphing entre las tres imágenes ejemplo con parámetros de seguimiento más cercanos a los estimados sobre la imagen actual del usuario y promediando sus valores de gris.

En el trabajo de Trevor Darrell [24] se emplean varias imágenes prototipo, de varias zonas de la cara, para las que se conocen los parámetros de animación que

generará una apariencia similar en un modelo 3D del rostro. El seguimiento, en el que únicamente se estima la traslación, se basa en la correlación normalizada de cada una de las imágenes ejemplo con las imágenes de entrada (para lo que emplean hardware especializado). La estimación de los parámetros de animación es el resultado de la interpolación de los parámetros asociados con cada prototipo, en base a los valores de correlación obtenidos en varias imágenes de la secuencia. Por tanto, la animación del modelo 3D se basa en la apariencia de las diferentes regiones del rostro.

El sistema desarrollado por Valente [96, 97, 85], estima los MPEG-4 FAPS que presenta el rostro del usuario. Tratan con un modelo 3D muy realista de la cara. Suponen una relación lineal entre la apariencia de los ojos o la boca y sus parámetros de animación. Generan un conjunto de imágenes que provienen de proyectar el modelo 3D con diferentes parámetros de animación, orientaciones y posiciones. A partir de esos datos generados con el modelo sintético se calcula el estimador lineal que relaciona la apariencia con los parámetros de animación. Para poder estimar los parámetros de animación en una imagen se necesitan las imágenes estabilizadas de ojos y boca. Para localizar los ojos y la boca se emplea un seguidor que usa 5 puntos junto con un filtro de Kalman para estimar la posición y orientación 3D de la cabeza.

Otro sistema basado en MPEG-4 se debe a Alhberg [1]. Emplea Modelos Activos de Apariencia (AAM) [21] para ajustar el modelo de alambres a la imagen. El algoritmo necesita de una estimación de la posición de la cara en la primera imagen. Trabaja a 5 cuadros por segundo y reconoce 6 de los 68 FAPS.

Para terminar el repaso por algunos sistemas que emplean MPEG-4 FAPS para codificar las expresiones faciales, hablaremos sobre el construido por Taro Goto [37]. Se extraen fonemas mediante reconocimiento de voz y se mezclan con la información visual. El modelo 3D del usuario (para emplearlo en vídeo-conferencia, por ejemplo) se construye a partir de 2 fotos: una de frente y otro de lado. En el seguimiento de las regiones de la cara se emplea información de los contornos de ojos, labios, cejas, barbilla y la posición de las pupilas. Con esta información consiguen extraer FAPS entre 10 y 15 cuadros por segundo en un PIII 500MHz con imágenes de 320x240 píxeles.

Otro grupo importante de sistemas de análisis facial son aquellos que emplean FACS (*Facial Action Parameters Set*) como sistema de codificación de expresiones faciales. En el trabajo de Cohn [18], se trata de distinguir los parpadeos y los movimientos de las cejas de forma automática. Para comenzar el seguimiento, hay que marcar 7 puntos de forma manual en la primera imagen (de la ceja y el ojo derecho) aunque a partir de ahí el sistema es automático. Se emplea un seguidor 3D para obtener una vista estabilizada de la cara (el modelo 3D es un cilindro). Los resultados obtienen un 100 % de aciertos con los parpadeos y un 57 % con los movimientos de las cejas. Para validar los resultados utilizan secuencias codificadas manualmente (por codificadores FACS expertos) en situaciones realistas: movimientos bruscos de la cara, movimientos fuera del plano, cambios de iluminación.

Terminaremos con el trabajo de Tian [89] que pertenece al grupo de sistemas que emplean FACS. Se emplean seguidores ad-hoc, basados en plantillas deformables con

estados múltiples (p. ej: una linea curva para boca apretada, una curva cerrada para boca cerrada y dos curvas cerradas concéntricas para la boca abierta) para estimar el estado de la boca, los ojos, las cejas, las mejillas y las arrugas nasolabiales. También se detecta la presencia de arrugas entre las cejas. Para saber si aparecen o no ciertos FACS en la imagen, se trata por separado la zona de los ojos (de la nariz para arriba) y la de la boca. Empleando dos redes neuronales (una para la región de los ojos y otra para la zona de la boca) establecen si aparecen el la imagen ciertos FACS. El vector de entrada a la red neuronal lo forman los parámetros de las distintas plantillas deformables. Los resultados se encuentran en torno al 96 % de aciertos funcionando a 2 imágenes por segundo. Las plantillas deformables se colocan manualmente sobre la cara en la primera imagen y a partir de ahí el funcionamiento es automático. Las pruebas del sistema se realizan con secuencias frontales a la cámara y representando unidades de acción (gestos) conocidos.

7.1.2. Animación dirigida por la voz

Una de las razones más frecuentes por las que una animación del rostro humano puede parecer irreal es que el movimiento de los labios no se corresponda con el sonido. Desde el punto de vista de las aplicaciones generar una animación del rostro a partir de la voz es muy atractivo. Principalmente porque el sistema de animación permitiría reutilizar toda la tecnología de transmisión de voz existente. Por otro lado, para construir una animación facial a partir de la voz es necesario procesar un número importante de secuencias de vídeo con voz e imagen sincronizados. Esto es así, porque es necesario aprender la relación entre conjuntos de fonemas y movimiento facial.

Cristoph Bregler desarrolló el primer sistema de animación vídeo-realista dirigida por la voz completamente automático [14]. El sistema, llamado *Video Rewrite*, etiqueta fonemas de tres en tres (trifonemas) en los vídeos de entrenamiento y de entrada. Después reordena las imágenes de entrenamiento para encajar con la secuencia de fonemas de la nueva locución en el vídeo de entrada. Si los fonemas de entrada no existen en los datos de entrenamiento, se eligen los más próximos. En *Video Rewrite* se siguen puntos de la boca del actor tanto en las secuencias de entrenamiento como en la secuencia de entrada y se emplean técnicas de "morphing" para mezclar las imágenes de la boca y el vídeo de entrada. Dada una secuencia de vídeo, una nueva locución y la base de datos de trifonemas resultado del entrenamiento (con sus imágenes correspondientes), se puede generar un nuevo vídeo con las maneras y el movimiento de la secuencia de entrada, moviendo la boca con la dinámica del entrenamiento y en sincronía con la nueva locución. El gran problema con la aproximación de Video Rewrite es la necesidad de almacenar los trifonemas (1700 extraídos a partir de 8 minutos de vídeo).

Matthew Brand, en su trabajo sobre animación dirigida por la voz [13], consigue animar una cara para que sea consistente con una locución incluso a partir de una única fotografía. El sistema se entrena a partir unas secuencia de imágenes con movimiento facial junto con el sonido. Se construyen Modelos Ocultos de Markov

(HMM) para el movimiento facial (que se obtiene del vídeo mediante el seguimiento de ciertos puntos sobre la cara) y para la voz. A posteriori, dada una nueva secuencia de fonemas se puede emplear el HMM del sonido para predecir los movimientos faciales y generar la trayectoria del nuevo movimiento facial.

Tony Ezzat por su parte, desarrolló un sistema de generación de animación del habla vídeo-realista mediante aprendizaje [29]. Para la generación de nuevas imágenes el sistema emplea un modelo, que recibe el nombre de *Multidimensional Morphable Model* o *MMM* (con una filosofía parecida a los Modelos Activos de Apariencia [21]). El MMM se tiene un conjunto de 46 imágenes prototipo para representar la textura junto con el flujo óptico entre una imagen referencia y cada uno de los prototipos para representar la forma. Este modelo emplea 46 parámetros para representar la forma y otros 46 para la textura. Una parte importante del trabajo lo constituye la técnica de síntesis de trayectorias en el espacio de parámetros del *MMM* que se entrena a partir de secuencias de vídeo grabadas. El generador de trayectorias permite generar secuencias de imágenes de la boca a partir de una secuencia de fonemas. El resultado final es un vídeo completamente creíble (como se demuestra en los experimentos [29]) en el que se colocan las imágenes de la boca generadas sobre una secuencia de vídeo real. El principal inconveniente es que se necesitan de 15 minutos de vídeo sincronizado con el audio, sabiendo qué fonemas se están pronunciando; varios días para entrenarse; y la orientación de la cabeza del hablante en el vídeo sintético es la misma que en el entrenamiento.

7.2. Reanimación

La filosofía que proponemos para realizar la animación de un modelo 3D de la cara es similar a la presentada por Valente y Dugelay [98, 85]. Valente y Dugelay, emplearon flujo óptico como característica discriminante de las expresiones faciales y un modelo 3D de la cara muy realista para cada usuario concreto. El modelo 3D les permitía obtener el flujo óptico correspondiente a cada zona de la cara sin más que modificar los parámetros de animación. Finalmente para obtener las imágenes estabilizadas de diferentes regiones de la cara, utilizaron un seguidor de características (cinco puntos) y un filtro de Kalman. Debido a que su seguidor no está diseñado para enfrentarse a movimiento no-rígido, no queda claro cómo funcionaría con expresiones las faciales más extremas (p.ej. al levantar las cejas o abrir la boca completamente).

En nuestro caso el seguidor basado en la apariencia del capítulo 6 nos permitirá localizar y seguir las partes más informativas de la cara a pesar del movimiento no-rígido. El resultado es que mediante su uso podremos extraer imágenes normalizadas de cualquier parte de la cara. Nuestra aproximación al problema una vez más se basará en el aprendizaje. En una etapa de entrenamiento previa al seguimiento se le mostrarán al sistema un conjunto de ejemplos, constituidos por imágenes y parámetros de animación asociados, y éste aprenderá a estimar los parámetros de animación a partir de las imágenes.

7.2.1. Relación lineal entre dos conjuntos de datos

El problema de la reanimación consiste en encontrar la relación entre los parámetros de seguimiento y los parámetros de animación. Supongamos que $\mathbf{D}_1$ es el conjunto de muestras con parámetros de seguimiento y $\mathbf{D}_2$ el conjunto de muestras con parámetros de animación. Resolver el problema de la reanimación supondrá que, dada una muestra $\bar{d}_1 \in \mathbf{D}_1$ podremos estimar cual es su correspondiente muestra $\bar{d}_2 \in \mathbf{D}_2$.

En esta sección trataremos el problema de la estimación de modelos lineales para representar dos conjuntos de datos multidimensionales (el de los parámetros de seguimiento y el de los parámetros de animación). Nuestro problema consiste en encontrar la relación entre dos conjuntos de datos, $\mathbf{D}_1 \in \mathcal{R}^{d_1 \times n}$ y $\mathbf{D}_2 \in \mathcal{R}^{d_2 \times n}$, con igual número de de muestras n, de forma que podamos estimar el elemento en un conjunto que se corresponde con un elemento en el otro. En la literatura se han empleado diferentes formas de encontrar la relación lineal entre dos conjuntos de datos acoplados [25]:

- **PCA simultáneo**. Consiste en construir una nueva matriz de datos $\mathbf{D}_2^1 = \begin{bmatrix}\mathbf{D}_1^\top & \mathbf{D}_2^\top\end{bmatrix}^\top$. Se hace PCA sobre la matriz aumentada, $\mathbf{D}_2^1$ y se obtiene la base del subespacio aumentado $\mathbf{B}_2^1 = \begin{bmatrix}\mathbf{B}_1^\top & \mathbf{B}_2^\top\end{bmatrix}^\top$. Para la predicción de un conjunto de datos a partir del otro (el caso asimétrico), esta aproximación no es óptima. Es importante destacar que el subespacio encontrado mediante el PCA conjunto es equivalente a minimizar:

  $$E(\mathbf{B}_1, \mathbf{B}_2, \mathbf{C}) = \sum_{i=1}^{n} ||d_1^i - \mathbf{B}_1 c_i||^2 + \sum_{i=1}^{n} ||d_2^i - \mathbf{B}_2 c_i||^2,$$

  donde los coeficientes son compartidos pero las bases son diferentes para cada conjunto de datos.

- **Reducción de dimensiones independiente**. Una forma diferente de abordar el problema consiste en reducir las dimensiones de cada conjunto de datos por separado. A continuación se aprende la relación entre los coeficientes de cada conjunto de datos en dimensiones reducidas (p.ej. mediante una red de neuronas). Esta aproximación puede suponer que ciertos elementos necesarios que influyen en la correlación de los dos conjuntos de datos se hayan eliminado en el paso previo de reducción de la dimensionalidad por separado.

- **PCA asimétrico**. La última forma de resolver el problema se basa en el PCA asimétrico:

  $$E(\mathbf{B}_1, \mathbf{B}_2, \mathbf{C}) = \sum_{i=1}^{n} ||d_{2,i} - \mathbf{B}_2 \mathbf{B}_1^\top d_{1,i}||^2.$$

  Por tanto los coeficientes son compartidos y se tratará de encontrar $\mathbf{B}_1, \mathbf{B}_2$ y $\mathbf{C}$.

Para nuestras pruebas sobre reanimación emplearemos el PCA simultáneo que se ha venido empleando en los Modelos Activos de Apariencia (AAM) para la forma y los niveles de gris con gran éxito.

7.2.2. Estimación de los parámetros de animación

Para estimar los parámetros de animación de una región de la cara partiremos de e imágenes ejemplo con N píxeles cada una. Sea $\mathbf{I}$ una matriz $N \times e$, donde cada columna $\bar{i}_k$ consta de los N píxeles de la región de interés (p.ej. recorriendo la imagen por filas), y e es el número de ejemplos que se van a emplear. Por otro lado, sea $\mathbf{A}$ una matriz $a \times e$, donde cada columna $\bar{a}_k$ presenta los a parámetros de animación asociados a la apariencia representada en $\bar{i}_k$ [1]. Por otro lado, sea $\mathbf{D}_r$ una matriz $(N+a) \times e$ de la forma:

$$\mathbf{D}_r = \begin{pmatrix} \mathbf{I} \\ \mathbf{W}_A\mathbf{A} \end{pmatrix} = \begin{pmatrix} \bar{i}_1 & \cdots & \bar{i}_e \\ \mathbf{W}_A[\bar{a}_1 & \cdots & \bar{a}_e] \end{pmatrix}, \tag{7.1}$$

donde $\mathbf{W}_A$ es una matriz diagonal de pesos que sirven para tener en cuenta las diferencias de escala entre los parámetros de animación y los niveles de gris. La matriz de pesos que empleamos es la matriz diagonal $r\mathbf{I}$ donde r^2 es la proporción entre la varianza total en niveles de gris y la varianza total en los parámetros de animación. En el caso de los Modelos de Directos de Apariencia (Direct Appearance Models) [42], se emplea una matriz de pesos análoga pero en este caso sobre niveles de gris y parámetros de forma.

Aplicando PCA sobre $\mathbf{D}_r$, obtendremos $\mathbf{B}_r$, base del subespacio expandido por los r autovectores asociados a los mayores autovalores de la matriz de covarianzas $(\mathbf{D}_r\mathbf{D}_r^\top)$, y que tendrá la forma

$$\mathbf{B}_r = \begin{pmatrix} \mathbf{B}_i \\ \mathbf{B}_a \end{pmatrix}. \tag{7.2}$$

Mediante $\mathbf{B}_r$, matriz $(N+a) \times r$, podremos encontrar un vector de r elementos, $\bar{c}_r$, que representará la correlación existente entre las imágenes en $\mathbf{I}$ y los parámetros de animación en $\mathbf{A}$. De forma que dada una nueva pareja $(\bar{i}, \bar{a})$ se podrá aproximar mediante $(\bar{i}^*,\bar{a}^*)$:

$$\bar{c}_r = \mathbf{B}_r^\top \begin{pmatrix} \bar{i} \\ \mathbf{W}_A\bar{a} \end{pmatrix} \tag{7.3}$$

$$\begin{pmatrix} \bar{i}^* \\ \mathbf{W}_A\bar{a}^* \end{pmatrix} = \mathbf{B}_r\bar{c}_r. \tag{7.4}$$

Lo que nos interesa es, dada una imagen de ejemplo $\bar{i}$, estimar sus parámetros de animación asociados, $\bar{a}^*$. Si conociésemos los parámetros de correlación, $\bar{c}_r$, para

[1] Supondremos a partir de aquí, y sin pérdida de generalidad, que todos los ejemplos $\bar{i}_k$ se encuentran centrados en la media al igual que se encuentran los parámetros de animación $\bar{a}_k$

una imagen $\bar{i}$ sus parámetros de animación asociados serían $\bar{a} = \mathbf{B}_a \bar{c}_r$. El problema entonces se convierte en la estimación de $\bar{c}_r$ dadas una imagen $\bar{i}$, y las matrices provenientes del entrenamiento $\mathbf{B}_i$ y $\mathbf{B}_a$.

Conociendo la estructura de $\mathbf{B}_r$ podemos escribir

$$\mathbf{B}_i \bar{c}_r = \bar{i}, \tag{7.5}$$

donde $\bar{c}_r$ es la única incógnita. En general, el número de píxeles de la imagen N es mucho mayor que r. Elegiremos la solución que minimiza

$$\bar{c}_r^* = \min_{\bar{c}_r} ||\mathbf{B}_i \bar{c}_r - \bar{i}||^2 = pinv(\mathbf{B}_i)\bar{i}, \tag{7.6}$$

donde la matriz de dimensiones $r \times N$, $pinv(\mathbf{B}_i)$, es la pseudo-inversa de $\mathbf{B}_i$ calculada empleando la Descomposición en Valores Singulares (SVD) [75, Sección 2.6].

El resultado es que los parámetros de animación correspondientes a la imagen $\bar{i}$ se pueden calcular como:

$$\mathbf{W}_A \bar{a}^* = \mathbf{B}_a pinv(\mathbf{B}_i)\bar{i} = \mathbf{R}_i^a \, \bar{i}, \tag{7.7}$$

donde $\mathbf{R}_i^a$ es una matriz, $a \times N$, que es constante y se puede precalcular. Por otro lado dado que obtenemos $\mathbf{W}_A \bar{a}^*$ tendremos que multiplicar el resultado por $(\mathbf{W}_A)^{-1}$ para obtener la estimación de los parámetros de animación en la escala adecuada, $\bar{a}^*$.

7.3. Experimentos

En esta sección expondremos los experimentos realizados para validar la aplicación de animación facial que hemos desarrollado. En primer lugar, las pruebas sintéticas con parámetros de animación conocidos que nos permitirán establecer el error cometido. Posteriormente se realizarán pruebas sobre secuencias de imágenes reales de varios usuarios diferentes para observar el comportamiento del sistema construido. En todos los casos trataremos la región de los ojos de forma separada a la región de la boca, ya que el movimiento de ambas es casi independiente (ver figura 7.3), lo que nos permitirá reducir el número de ejemplos necesarios en el entrenamiento (siguiendo el espíritu del PCA modular del que hablamos en el capítulo 6).

7.3.1. Validación cuantitativa

En el primer experimento queremos validar cuantitativamente la calidad de la reanimación. Para ello utilizamos el modelo gráfico 3D del rostro del apéndice C para generar tres secuencias de imágenes:

- **Entrenamiento de los ojos**. En la primera secuencia, de 630 imágenes, hacemos que la cabeza permanezca estática (sin rotaciones o translaciones globales) y que sólo presente movimiento no rígido en la zona de los ojos (ver figura 7.4).

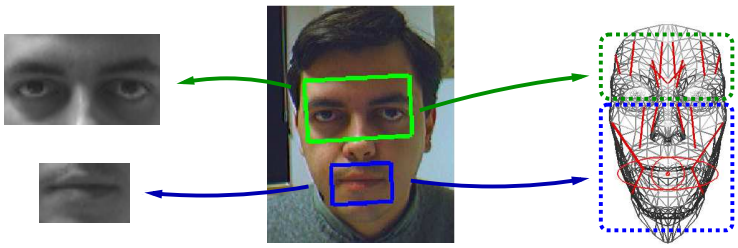


Figura 7.3: Separación entre boca y ojos en el análisis de expresiones faciales. A la izquierda imágenes estabilizadas de boca y ojos. A la derecha del todo, modelo 3D del rostro con músculos implicados en cada área.

- **Entrenamiento de la boca**. En la segunda secuencia, de 540 imágenes, hacemos que la cabeza permanezca estática (sin rotaciones o translaciones globales) y que sólo presente movimiento no rígido en la zona de la boca (ver figura 7.5).
- **Prueba de la reanimación**. La tercera secuencia consiste en 1225 imágenes con movimientos tanto rígidos como no rígidos del rostro. Las expresiones faciales que ahora utilizaremos ahora no coinciden con las que se utilizaron en el entrenamiento (ver figura 7.6).

Procesando las dos secuencias de imágenes de entrenamiento mediante el procedimiento automático presentado en el apéndice B, tendremos el modelo de PCA para los ojos y la boca, así como las correspondientes imágenes estabilizadas (ver figuras 7.8, 7.10 y 7.11). Dado que las tres secuencias se generaron con el modelo del apéndice C, conocemos los valores de los parámetros de animación en cada imagen. Las imágenes estabilizadas junto con los parámetros de animación conocidos nos permiten estimar la matriz de reanimación, $\mathbf{R}_a^i$, para las dos regiones.

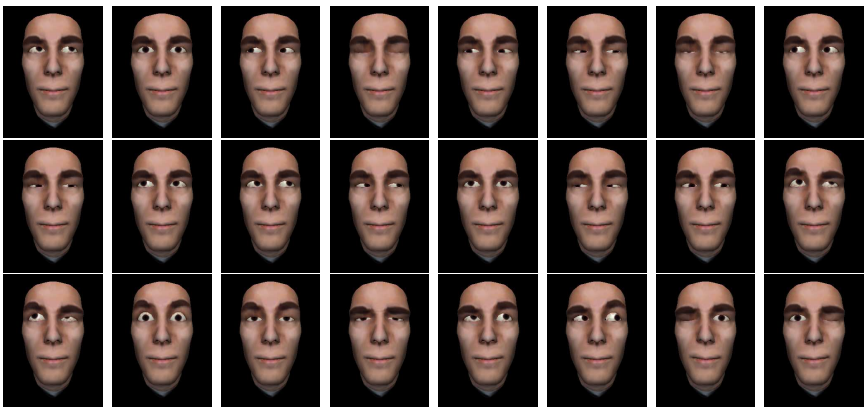


Figura 7.4: Expresiones faciales del modelo 3D para el entrenamiento del seguidor de la apariencia y la reanimación de los ojos. Se generó una secuencia de 630 imágenes interpolando entre las 24 expresiones mostradas en la figura.

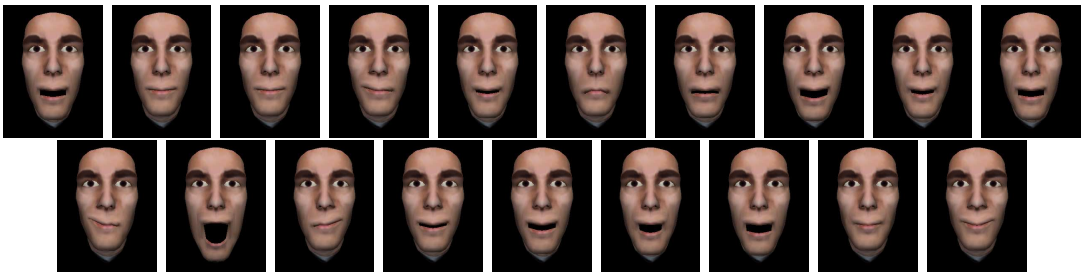


Figura 7.5: Expresiones faciales del modelo 3D para el entrenamiento del seguidor de la apariencia y la reanimación de la boca. Se generó una secuencia de 540 imágenes interpolando entre las 18 expresiones mostradas en la figura.

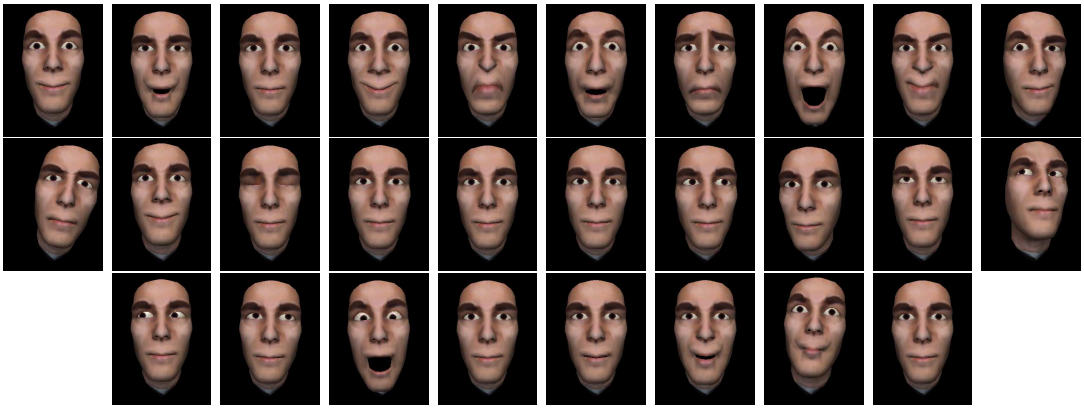


Figura 7.6: Expresiones faciales del modelo 3D para las pruebas del de la reanimación. Se generó una secuencia de 1225 imágenes interpolando entre las 75 expresiones. En la figura se muestran 28 de ellas.

El modelo gráfico de animación del rostro tiene 32 grados de libertad (ver apéndice C) aunque en este experimento dejaremos de lado la rotación y traslación 3D centrándonos en los parámetros que producen el movimiento no rígido en el modelo.

En el experimento cuantitativo queremos probar combinaciones de una serie de variables, y establecer cómo afectan a la calidad de los parámetros estimados. A tal fin se utilizaran diferentes:

- **Modelos de movimiento** en el seguimiento basado en la apariencia: rotación-traslación-escala (RTE), afín y proyectivo.
- **Tamaños de la región de la boca** en el seguimiento basado en la apariencia y reanimación: 53×43 (ver figura 7.11) y 35×23 píxeles (ver figura 7.10).
- **Particiones de la región de los ojos.** Una única región de 60×35 píxeles o dos de 30×35 píxeles cada una.
- **Conjuntos de ejemplos 2D-3D** para estimar la relación entre los niveles de gris y los parámetros de animación: la matriz $\mathbf{R}_i^a$. En el caso de la boca, "pocos ejemplos" quiere decir 18 (ver figura 7.9), y "muchos ejemplos" quiere decir 540. Para los ojos, en cambio, "pocos ejemplos" son 21 y "muchos ejemplos" son 629 (ver figura 7.7). Los ejemplos del grupo de "pocos ejemplos", tanto para la boca como para los ojos, se han elegido manualmente intentando cubrir un rango de expresiones faciales suficiente para producir una animación expresiva.

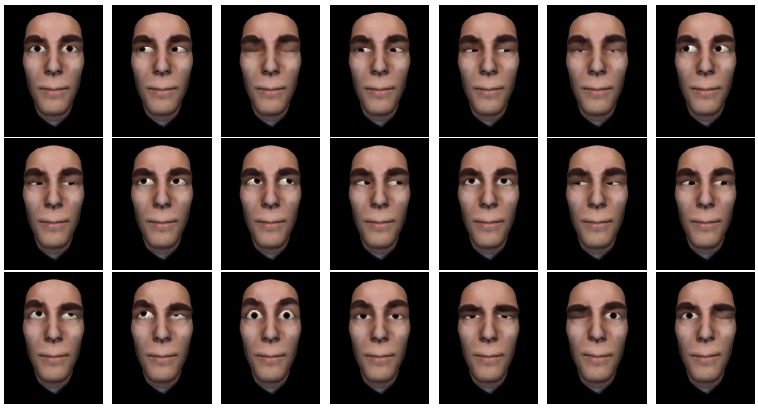


Figura 7.7: Mínimo número de expresiones de ejemplo (21) elegidas para realizar el entrenamiento de los ojos para la reanimación (empleados cuando se habla de "pocos ejemplos").

Con este experimento tratamos de seleccionar el mejor estimador de los parámetros de animación posible, el mejor, en el sentido de comportarse mejor en los peores casos. De esta forma, la secuencia de prueba presenta las codiciones de trabajo menos favorables a nuestro algoritmo: expresiones faciales que no aparecen entre las de entrenamiento y movimientos fuera de la frontal a la cámara. Así que se trata de una prueba de "peor caso".

Figura 7.8: Mínimo número de imágenes de ejemplo (21) elegidas para realizar el entrenamiento de los ojos para la reanimación (empleados cuando se habla de "pocos ejemplos").

Figura 7.9: Mínimo número de expresiones ejemplo de la boca elegidas para realizar el entrenamiento para la reanimación (empleados cuando se habla de "pocos ejemplos").

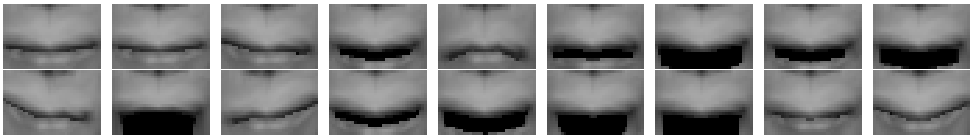


Figura 7.10: Mínimo número de imágenes ejemplo de la región de la boca, de 35×23 píxeles, elegidas para realizar el entrenamiento para la reanimación (empleados cuando se habla de "pocos ejemplos").

Figura 7.11: Mínimo número de imágenes ejemplo de la región de la boca, de 53×43 píxeles, elegidas para realizar el entrenamiento para la reanimación (empleados cuando se habla de "pocos ejemplos").

En la figura 7.12 se muestran los resultados de las diferentes pruebas realizadas. En el eje vertical de cada gráfica tenemos el RMS (raíz cuadrada de la media de los errores al cuadrado) de cada parámetro de animación dividido por la longitud de su rango de posibles valores. De esta forma podemos comparar todos los parámetros teniendo en cuenta la precisión de su estimación. En el eje horizontal de cada gráfica aparecen primero los parámetros de la zona de los ojos y después los de la zona de la boca (ver cuadro 7.1). Las variaciones más significativas en las gráficas se producen en la estimación de los parámetros del ojo izquierdo (índices 1, 2 y 3), del ojo derecho (índices 9, 10 y 11) y de la apertura de la mandíbula (índice 17). En el caso de la apertura de la mandíbula el error de estimación se reduce conforme pasamos de emplear un modelo de movimiento RTE al proyectivo, si bien es cierto que no baja del 25 %. En el caso de la apertura de los párpados también se reduce el error al pasar del modelo RTE al proyectivo aunque el error final depende del número de ejemplos empleados. Con un número grande de ejemplos tenemos un error del 15 % (con el modelo proyectivo) mientras que es de un 20 % (con el modelo proyectivo) cuando se emplea un número mucho más reducido de ejemplos. Podemos concluir que empleando el modelo de movimiento proyectivo tendremos la estimación más precisa de la apertura de la mandíbula y de los párpados. Si seguimos con las gráficas de la figura 7.12 se observa que la estimación de la orientación de los ojos (índices 2 y 3 para el ojo izquierdo, 10 y 11 para el derecho) presenta un error de entre 10 % y 20 % del rango total, o lo que es lo mismo, entre 9 y 18 grados. Por otro lado, el hecho de emplear una única región (AO) frente a dos (OS) en la zona de los ojos, supone que el error de estimación de la apertura de los párpados se puede reducir hasta en un 10 % y el de la orientación del ojo hasta en un 5 % (modelo RTE con pocos ejemplos). En general, el error en la estimación de la apertura de los párpados es menor empleando una única región en la zona de los ojos (AO), probablemente porque un mayor número de píxeles proporciona más información a la hora de calcular los parámetros de animación.

Hemos visto que el RMS proporcional al rango de cada parámetro de animación nos aporta información importante sobre la calidad de la reanimación. Ahora observaremos la evolución de la estimación de algunos parámetros frente al valor real de los mismos. En la figura 7.13 se presenta el resultado de la reanimación de los ojos con el modelo proyectivo y una única región, mientras que en la figura 7.14 se presentan los resultados empleando una región para cada ojo.

Las peores condiciones para nuestro algoritmo: movimientos rápidos fuera de la frontal a la cámara (entre las imágenes 222 y 435, 700 y 875) y expresiones faciales que no se encontraban entre las de entrenamiento (entre las imágenes 80 y 221) provocan un incremento del error en la estimación de los parámetros. Se llega a un error del 43 % (del rango de posibles valores del parámetro) en la apertura de los párpados en el caso de una única región para los ojos y un 57 % con dos regiones. En el caso de la orientación de los ojos el máximo error es del 10 %, con una única región, y del 20 %, con una región para cada ojo. Sin embargo, cuando tenemos condiciones favorables a nuestro algoritmo (movimientos no rígidos en posición frontal a la cámara entre las imágenes 436 y 700) el máximo error en la apertura de los

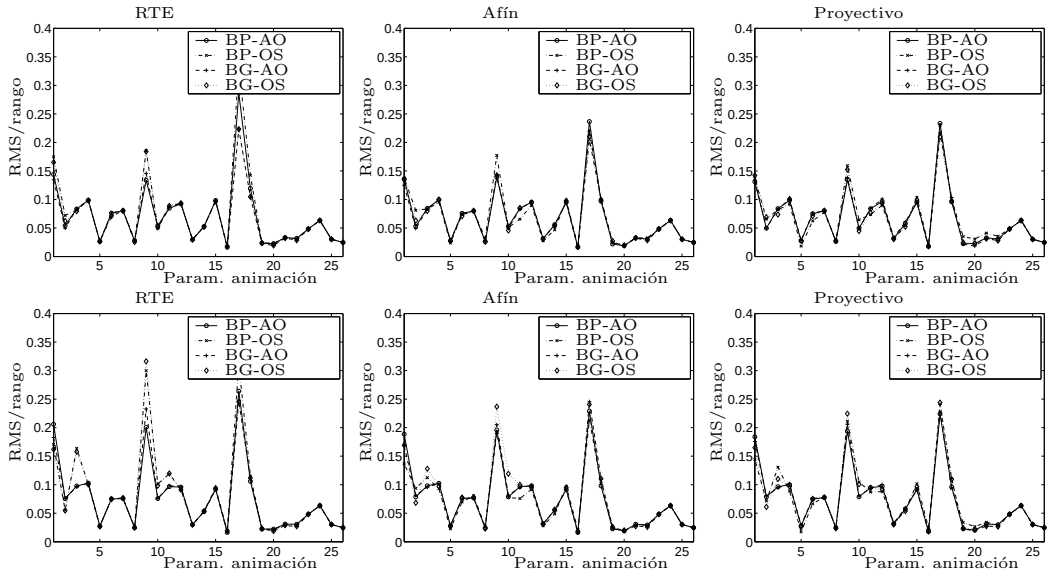


Figura 7.12: Valores de RMS (Root-Mean-of-Squares) para los parámetros de animación en las diferentes pruebas dividido por la longitud de su rango de posibles valores. De izquierda a derecha resultados para el modelo de movimiento RTE, Afín y Proyectivo. Fila de arriba, resultados empleando 629 y 540 ejemplos para la boca y los ojos, respectivamente, en la reanimación. Fila de abajo, resultados empleando 21 y 18 ejemplos para la boca y los ojos, respectivamente, en la reanimación. En las gráficas se comparará el uso de una región grande (BG) de 53×43 píxeles con una región más pequeña de la boca (BP) de 35×23. También se compara el uso de una única región con ambos ojos (AO) de 60×35 píxeles o dos regiones separadas para los ojos (OS) de 30×35 píxeles cada una.

Parámetros ojos	índice
Apertura párpado izquierdo	1
Rot. vertical ojo izquierdo	2
Rot. horizontal ojo izquierdo	3
Frontal interior izquierdo (músculo)	4
Frontal mayor izquierdo (músculo)	5
Frontal externo izquierdo (músculo)	6
Depresor de ceja izquierda (músculo)	7
Secundario frontal izquierdo (músculo)	8
Apertura párpado derecho	9
Rot. vertical ojo derecho	10
Rot. horizontal ojo derecho	11
Frontal interior derecho (músculo)	12
Frontal mayor derecho (músculo)	13
Frontal externo derecho (músculo)	14
Depresor de ceja derecha (músculo)	15
Secundario frontal derecho (músculo)	16
Parámetros boca	**índice**
Apertura mandíbula	17
Contracción labios (músculo)	18
Cigomático mayor derecho (músculo)	19
Cigomático mayor izquierdo (músculo)	20
Depresor angular derecho (músculo)	21
Depresor angular izquierdo (músculo)	22
Labio-nasal derecho (músculo)	23
Labio-nasal izquierdo (músculo)	24
Labio-nasal interior derecho (músculo)	25
Labio-nasal interior izquierdo (músculo)	26

Cuadro 7.1: Índices de los diferentes parámetros de animación.

párpados se encuentra alrededor del 7 % y en la orientación de los ojos alrededor del 3 % independientemente del número de regiones empleadas en el seguimiento de los ojos. La conclusión que podemos obtener es que empleando una única región tendremos un error menor en la estimación de los parámetros cuando las condiciones no son las óptimas para nuestro algoritmo.

Por otro lado, en las imágenes 494 y 630 se producen sendos parpadeos lo que provoca la aparición de dos picos en las gráficas de ambos párpados (el valor 1 representa al ojo completamente cerrado). La forma muy apuntada de los picos en las gráficas nos indica que sería posible detectar los parpadeos observando la derivada del parámetro de apertura de los párpados.

Una situación importante que marca la diferencia entre utilizar una o dos regiones es el comportamiento cuando los ojos se mueven de forma independiente (alrededor de la imagen 1050 los ojos convergen y el modelo se pone bizco). En

este último caso, cuando empleamos una única región en la zona de los ojos (ver figura 7.13) y dado que no aparecía la expresión "bizco" entre las de entrenamiento de la reanimación, la estimación de las rotaciones en horizontal de ambos ojos (alrededor del eje vertical del ojo) es errónea ya que ambos presentan el mismo ángulo (y no con signos diferentes como debiera ser). Sin embargo, cuando se emplean dos regiones para los ojos (ver figura 7.14), sí que se estiman correctamente los ángulos. A pesar de que la expresión no se encontraba entre las de entrenamiento, al emplear dos regiones, cada ojo puede adoptar un ángulo diferente en la reanimación. Por otro lado, un aspecto muy importante de la reanimación es que esta sea creíble. No hay nada menos creíble que una persona que no mueva los dos ojos siempre en la misma dirección (independientemente de que ocasionalmente se ponga bizca) y eso es precisamente lo que ocurre en el caso de emplear dos regiones para los ojos. Como se puede observar en la figura 7.14 las rotaciones en vertical (alrededor del eje horizontal del ojo) y las horizontales (alrededor del eje vertical del ojo) difieren siempre incluso a veces de forma exagerada, lo que se traduce en una apariencia irreal del modelo.

En la figura 7.15 se presenta el resultado de la reanimación para la boca grande, mientras que en la figura 7.16 se presentan los resultados empleando la región pequeña de la boca. Igual que en el caso de los ojos nos interesa animar el modelo 3D del rostro de la forma más realista posible. Si la boca se abre y cierra de forma no sincronizada con el movimiento del usuario, la animación resultante no resulta creíble. La estimación de la apertura de la mandíbula obtenida empleando la región grande de la boca, únicamente es manifiestamente incorrecta alrededor de la imagen 850 (ver figura 7.15), lo que coincide con movimientos de la cabeza fuera de la frontal a la cámara. Sin embargo, cuando se emplea la región pequeña de la boca, la mandíbula aparece abierta en casi toda la secuencia, a pesar de que únicamente debería estarlo en tres ocasiones (ver figura 7.16). Algo muy parecido ocurre con la contracción del esfínter de los labios por lo que podemos concluir que es mejor emplear la región grande de la boca en la reanimación.

Las conclusiones que obtenemos de los experimentos cuantitativos son las siguientes:

- Deberemos emplear el mayor número de ejemplos para la reanimación para reducir el error.

- Emplear el modelo de movimiento proyectivo, dado que se ajusta mejor a las deformaciones producidas por el movimiento que el RTE o el afín, reduce el RMS en la estimación de los parámetros de animación.

- Elegir la región de la boca de forma que cubra la mayor área posible de la misma en el mayor número de expresiones faciales, mejora la precisión en la estimación de sus parámetros de animación.

- El seguimiento y animación de los dos ojos por separado es muy interesante porque se pueden representar configuraciones de los mismos que no estaban en

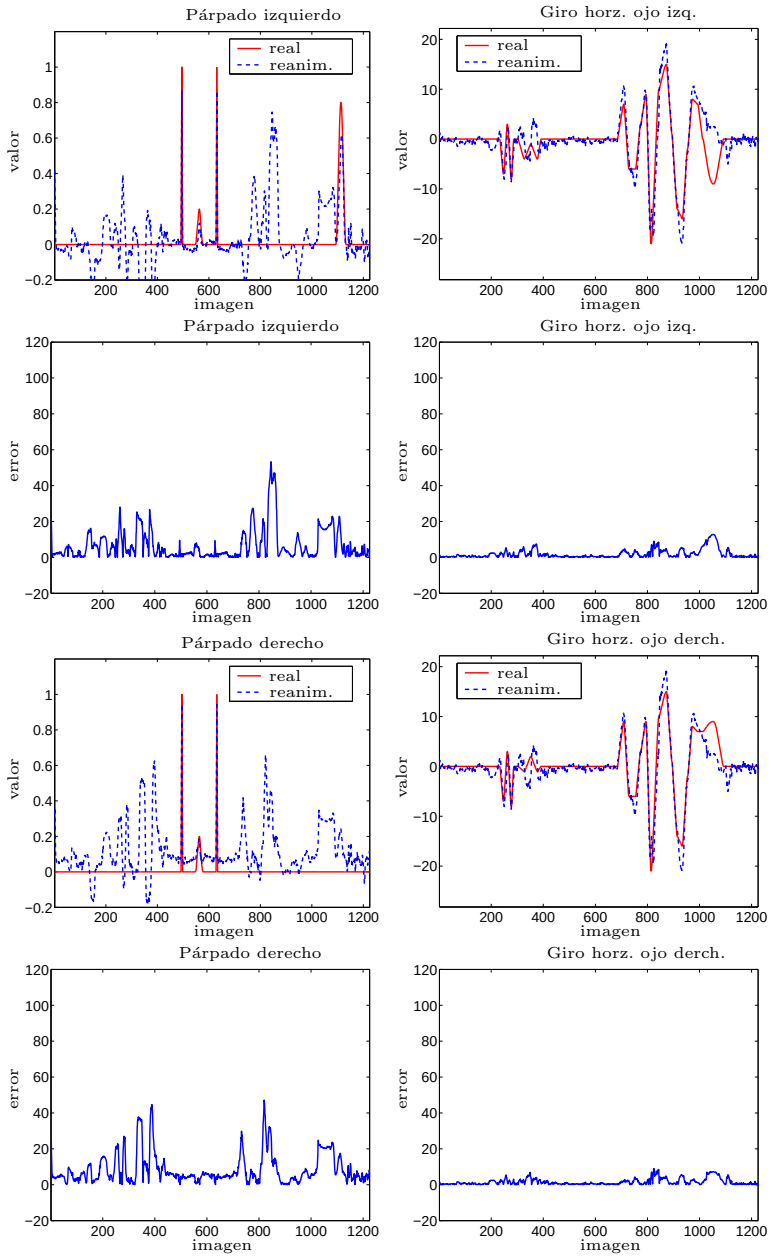


Figura 7.13: Parámetros de animación estimados para los ojos frente a los reales. Los datos son para el modelo de movimiento proyectivo, muchos ejemplos en el entrenamiento de la reanimación, boca grande y una única región para los ojos. En la primera fila se muestra la apertura del párpado junto a la orientación horizontal de la pupila del ojo izquierdo. En la tercera fila se muestran los errores correspondientes a los parámetros de la primera fila, medidos en porcentaje del rango de posibles valores del parámetro. En la tercera fila se muestra los mismos parámetros de la primera fila para el ojo derecho. En la cuarta fila se muestran los errores correspondientes a los parámetros de la tercera fila, medidos en porcentaje del rango de posibles valores del parámetro.

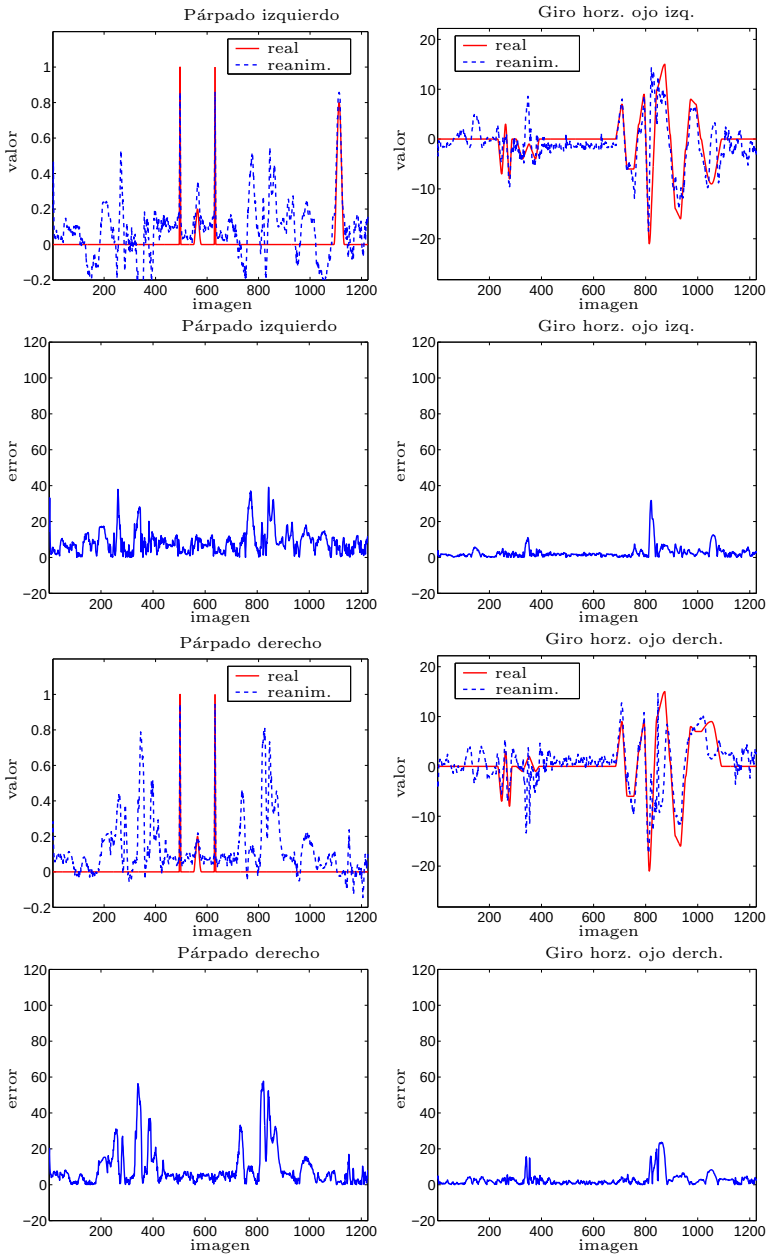


Figura 7.14: Parámetros de animación estimados para los ojos frente a los reales. Los datos son para el modelo de movimiento proyectivo, muchos ejemplos en el entrenamiento de la reanimación, boca grande y dos regiones para los ojos. En la primera fila se muestra la apertura del párpado junto a la orientación horizontal de la pupila del ojo izquierdo. En la tercera fila se muestran los errores correspondientes a los parámetros de la primera fila, medidos en porcentaje del rango de posibles valores del parámetro. En la tercera fila se muestra los mismos parámetros de la primera fila para el ojo derecho. En la cuarta fila se muestran los errores correspondientes a los parámetros de la tercera fila, medidos en porcentaje del rango de posibles valores del parámetro.

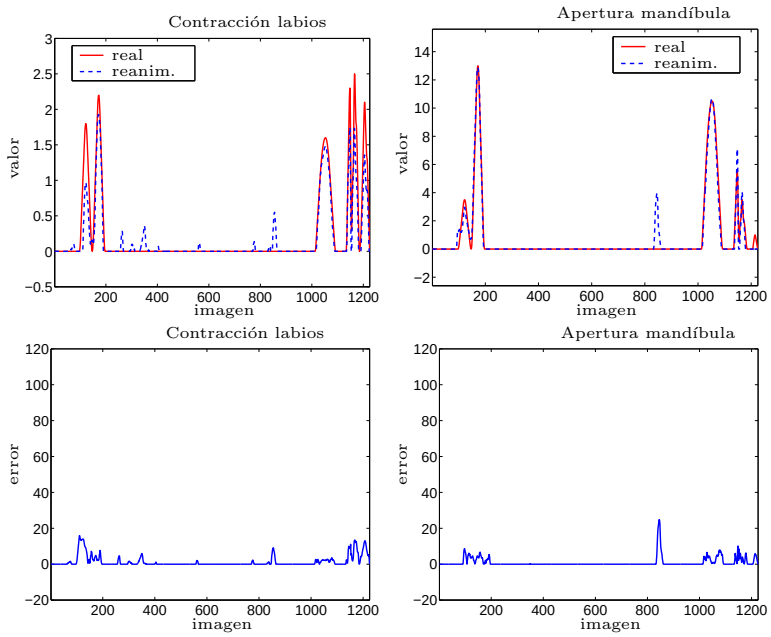


Figura 7.15: Parámetros de animación estimados para la boca frente a los reales. Los datos son para el modelo de movimiento proyectivo, muchos ejemplos en el entrenamiento de la reanimación, boca grande y una única región para los ojos. En la primera fila Se muestra el valor de contracción de los labios y la apertura de la mandíbula. En la segunda fila se muestran los errores correspondientes a los parámetros de la primera fila medidos en porcentaje del rango de posibles valores del parámetro.

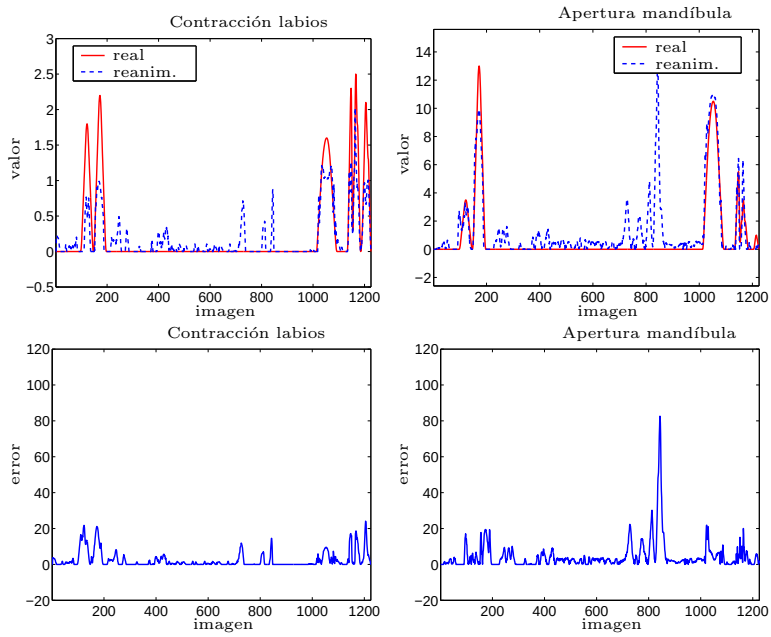


Figura 7.16: Parámetros de animación estimados para la boca frente a los reales. Los datos son para el modelo de movimiento proyectivo, muchos ejemplos en el entrenamiento de la reanimación, boca pequeña y una única región para los ojos. En la primera fila se muestra el valor de contracción de los labios y la apertura de la mandíbula. En la segunda fila se muestran los errores correspondientes a los parámetros de la primera fila medidos en porcentaje del rango de posibles valores del parámetro.

los ejemplos de entrenamiento. Sin embargo, en muchas ocasiones dos regiones provocan que los ojos aparezcan mirando en direcciones diferentes en el modelo 3D lo que no es nada realista. Una posible solución es trabajar con los dos ojos por separado y establecer un modelo de movimiento superior (un filtro de Kalman, por ejemplo) que haga que los ojos se muevan conjuntamente, aunque deje margen para que se puedan oponer (ponerse "bizcos").

- La estimación de las expresiones faciales es tanto mejor cuanto más frontal a la cámara se encuentra el usuario. Cuando no es así, el ruido (oscilaciones) en las estimaciones aumenta en gran medida. Una cuestión abierta es el filtrado de los parámetros de animación.

7.3.2. Validación cualitativa

En este apartado realizaremos procesaremos cuatro secuencias reales correspondientes a diferentes individuos (experimentos A, B, C y D). De acuerdo con los experimentos del apartado anterior, para los experimentos sobre secuencias de imágenes reales, elegiremos: el 85 % de variabilidad en el modelo de PCA para el seguimiento basado en la apariencia, una única región para los ojos, la región de la boca más

grande y el modelo de movimiento proyectivo.

El único problema aquí es la elección del número de ejemplos para la reanimación. Al tratarse de secuencias reales no conocemos los parámetros de animación del modelo adecuados para cada imagen estabilizada. La aproximación que hemos adoptado consiste en emplear los conjuntos de mínimo número de ejemplos de configuraciones del modelo 3D del apartado anterior (ver figuras 7.7 y 7.9) y encontrar por inspección visual cuales de las imágenes reales les corresponden. De esta forma tendríamos el conjunto de pares (imagen, parámetros de animación) necesarios en la fase de entrenamiento de la reanimación.

Experimento A

En el experimento A empleamos una secuencia de 4421 imágenes para el entrenamiento del modelo de apariencia y para las pruebas. Dividimos las imágenes en dos secuencias una de entrenamiento, 2360 imágenes, y una de prueba, 2061 imágenes. A partir de la secuencia de entrenamiento y siguiendo el procedimiento del apéndice B se entrenó el modelo para el seguimiento basado en la apariencia. De las imágenes estabilizadas de ojos y boca elegimos los ejemplos que se emplearán en el entrenamiento para la reanimación el cálculo de la matriz $\mathbf{R}_a^i$ (ver figuras 7.17 y 7.18).

Figura 7.17: Ejemplos de reanimación para la zona de los ojos en el experimento A.

Figura 7.18: Ejemplos de reanimación para la zona de la boca en el experimento A.

En la secuencia del experimento A hasta la imagen 360 el usuario sólo realiza expresiones faciales sin movimiento rígido. A partir de ahí, y hasta el final, el movimiento de la cabeza no cesa en ningún momento, incluso fuera de la frontal a la cámara. Aunque para apreciar la calidad de la reanimación sería necesario visualizar un vídeo o al propio sistema en funcionamiento, podemos hacernos una idea del comportamiento de los algoritmos desarrollados con las imágenes que se muestran en la figura 7.19.

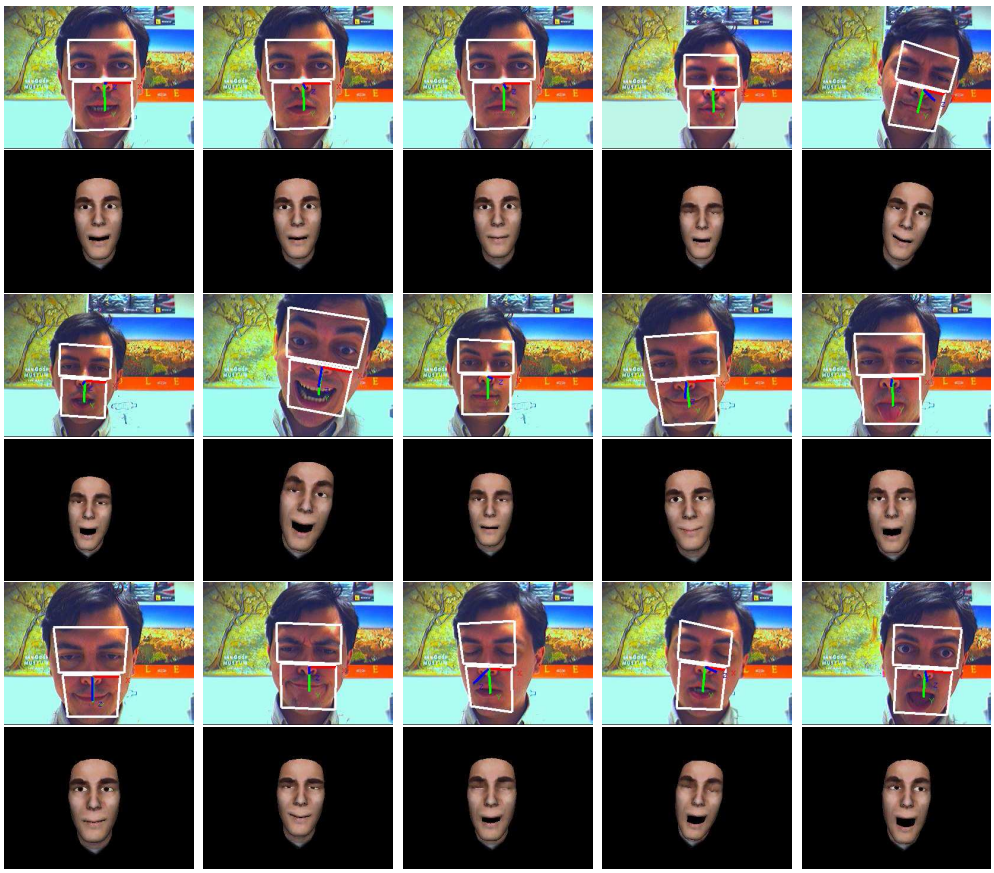


Figura 7.19: Resultados del experimento A. En las filas impares tenemos los resultados del seguimiento basado en la apariencia proyectivo. En las filas pares se muestran los resultados de la animación correspondientes con la fila inmediatamente superior. En este caso, se han eliminado de la reanimación las rotaciones alrededor del eje horizontal y alrededor del eje vertical de la cabeza para apreciar mejor las expresiones faciales. De izquierda a derecha y de arriba a abajo aparecen los resultado para las imágenes 100, 200, 300, 500, 600, 700, 800, 1100, 1200, 1300, 1400, 1500, 1700, 1900 y 2000.

Experimento B

En este experimento empleamos una secuencia de 4108 imágenes para el entrenamiento del modelo de apariencia y para las pruebas. Dividimos las imágenes en dos secuencias una de entrenamiento, 2375 imágenes, y una de prueba, 1733 imágenes. En la fase de entrenamiento para la reanimación (el cálculo de la matriz $\mathbf{R}_i^a$) se emplearon las imágenes de ejemplo que aparecen en las figuras 7.20 y 7.21.

Figura 7.20: Ejemplos de reanimación para la zona de los ojos en el experimento B.

Figura 7.21: Ejemplos de reanimación para la zona de la boca en el experimento B.

La secuencia del experimento B comienza con el usuario en el centro de la imagen realizando movimientos rígidos suaves de la cabeza junto con movimientos no rígidos (levantar las cejas, mover los ojos, hablar), imágenes 1 a 214. Los eventos más interesantes de la secuencias son giros hacia la izquierda de la imagen, y fuera de la frontal a la cámara, de la cabeza del usuario: imágenes 573 a 662, 776 a 885, 989 a 1080 y 1525 a 1568. También se produce un giro hacia la derecha de la imagen entre las imágenes 663 a 775. En todos estos giros el seguimiento proyectivo de la apariencia es capaz de continuar el seguimiento. Por último, a partir de la imagen 1669 el usuario se mueve rápidamente y el sistema no es capaz de continuar el seguimiento de la apariencia, lo hace con el seguidor basado en el color, con lo que la estimación de los parámetros de animación es incorrecta. Podemos hacernos una idea del comportamiento de los algoritmos desarrollados con las imágenes que se muestran en la figura 7.22.

Experimento C

En el tercer experimento con imágenes reales, utilizamos una secuencia de 4558 imágenes para el entrenamiento del modelo de apariencia y para las pruebas. Dividimos las imágenes en dos secuencias una de entrenamiento, 2745 imágenes, y una de prueba, 1813 imágenes. En la fase de entrenamiento para la reanimación se utilizaron las imágenes de ejemplo que aparecen en las figuras 7.23 y 7.24.

Figura 7.22: Resultados del experimento B. En las filas impares tenemos los resultados del seguimiento basado en la apariencia proyectivo. En las filas pares se muestran los resultados de la animación correspondientes con la fila inmediatamente superior. En este caso, se han eliminado de la reanimación las rotaciones alrededor del eje horizontal y alrededor del eje vertical de la cabeza para apreciar mejor las expresiones faciales. De izquierda a derecha y de arriba a abajo aparecen los resultado para las imágenes 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400 y 1500.

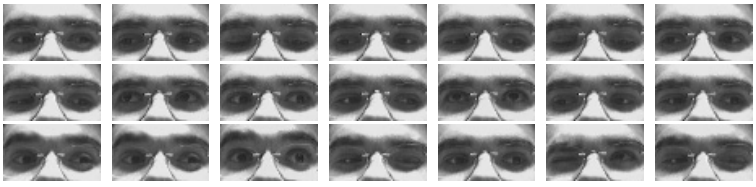


Figura 7.23: Ejemplos de reanimación para la zona de los ojos en el experimento C.

Figura 7.24: Ejemplos de reanimación para la zona de la boca en el experimento C.

En esta secuencia los movimientos de rotación fuera de la frontal a la cámara son constantes junto con movimientos rápidos de la cabeza. El resultado es que en multitud de ocasiones el seguidor de la apariencia se pierde. Podemos hacernos una idea del comportamiento de los algoritmos desarrollados con las imágenes que se muestran en la figura 7.25.

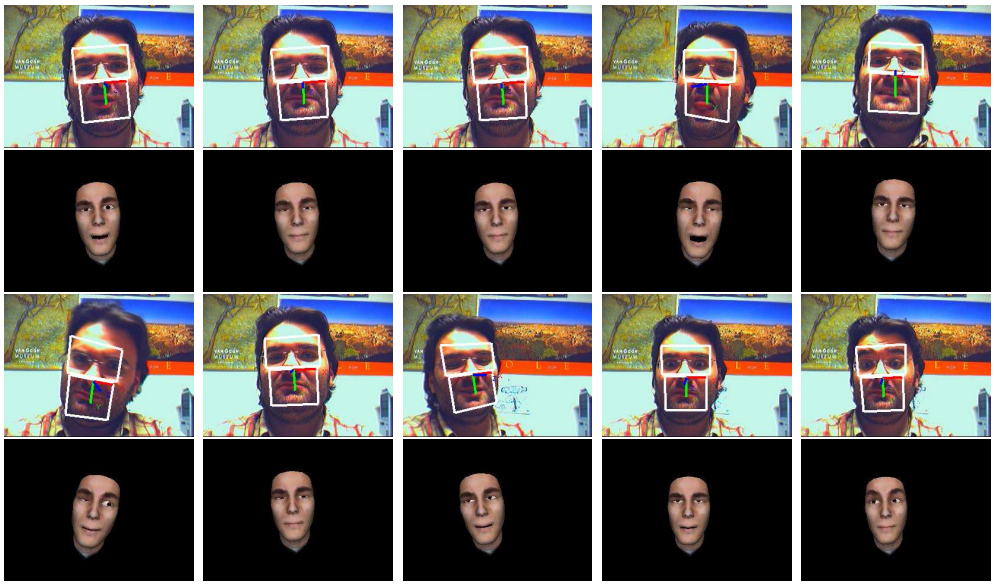


Figura 7.25: Resultados del experimento C. En las filas impares tenemos los resultados del seguimiento basado en la apariencia proyectivo. En las filas pares se muestran los resultados de la animación correspondientes con la fila inmediatamente superior. En este caso, se han eliminado de la reanimación las rotaciones alrededor del eje horizontal y alrededor del eje vertical de la cabeza para apreciar mejor las expresiones faciales. De izquierda a derecha y de arriba a abajo aparecen los resultado para las imágenes 100, 200, 300, 500, 700, 800, 1000, 1400, 1600 y 1700.

Experimento D

En el último experimento con imágenes reales, utilizamos una secuencia de 4925 imágenes para el entrenamiento del modelo de apariencia y para las pruebas. Dividimos las imágenes en dos secuencias, una de entrenamiento, 2190 imágenes, y una de prueba, 2735 imágenes. En la fase de entrenamiento para la reanimación se utilizaron las imágenes de ejemplo que aparecen en las figuras 7.26 y 7.27.

Figura 7.26: Ejemplos de reanimación para la zona de los ojos en el experimento D.

Figura 7.27: Ejemplos de reanimación para la zona de la boca en el experimento D.

En esta secuencia predominan las expresiones faciales que no se encuentran entre las de entrenamiento. Por tanto la estimación de los parámetros de animación e incluso el seguimiento basado en la apariencia no es muy preciso. Podemos hacernos una idea del comportamiento de los algoritmos desarrollados con las imágenes que se muestran en la figura 7.28.

7.4. Conclusiones

En este capítulo hemos mostrado una de las posibles aplicaciones del análisis facial: la animación dirigida por la actuación de una persona. Empleando un modelo gráfico 3D del rostro con 32 grados de libertad, podríamos implementar un sistema de vídeo-conferencia. Combinando el análisis facial en un extremo de la comunicación y la animación del modelo gráfico 3D en el otro extremo, necesitaríamos transmitir únicamente 32 parámetros por imagen (si suponemos 32 bits por número real sólo necesitaríamos 30Kbits/s para transmitir 30 imágenes por segundo).

El sistema de animación, mediante un entrenamiento previo, se puede adaptar a cualquier usuario y condiciones de iluminación. De esta forma, y tal como se ha demostrado en los experimentos, puede funcionar con cualquier usuario. Dado que la reanimación no supone más que la multiplicación de la matriz $\mathbf{R}_i^a$ correspondiente a cada región y las correspondientes imágenes estabilizadas, permite la reanimación del modelo 3D en tiempo real.

El sistema desarrollado, sin embargo, posee algunas limitaciones:

- La adaptación del sistema a una persona en concreto (el entrenamiento de la reanimación) es en parte manual. Esto es así debido a dos problemas abiertos: 1) no se ha estudiado cómo elegir el mejor conjunto de n expresiones faciales sobre el modelo 3D para la reanimación, por lo que se hace manualmente y 2) tampoco se ha estudiado cómo se pueden elegir automáticamente las imágenes de un usuario que se corresponden con cada una de las expresiones faciales del

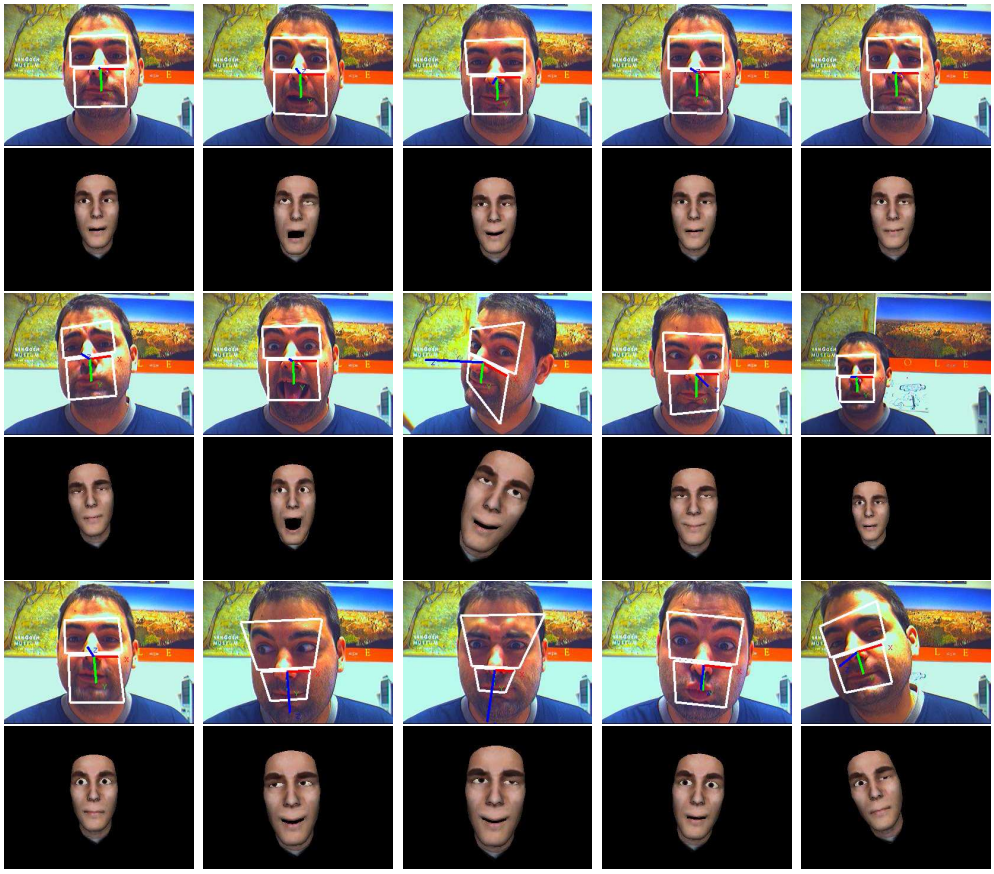


Figura 7.28: Resultados del experimento D. En las filas impares tenemos los resultados del seguimiento basado en la apariencia proyectivo. En las filas pares se muestran los resultados de la animación correspondientes con la fila inmediatamente superior. En este caso, se han eliminado de la reanimación las rotaciones alrededor del eje horizontal y alrededor del eje vertical de la cabeza para apreciar mejor las expresiones faciales. De izquierda a derecha y de arriba a abajo aparecen los resultado para las imágenes 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400 y 1500.

modelo 3D, y por tanto también se realiza este proceso manualmente. Sin embargo sí que se ha desarrollado un procedimiento automático para extraer las imágenes estabilizadas de cada región de la cara para cada nuevo usuario (ver apéndice B).

- El seguidor basado en la apariencia no es robusto ante cambios en las condiciones de iluminación. Y por tanto no lo es el sistema de animación que empleamos.
- El seguidor basado en la apariencia no es robusto ante oclusiones parciales. Ante una oclusión la estimación de los parámetros de animación es incorrecta.
- Se asume en todo momento que la cara se encuentra sobre un plano. Aunque el modelo de movimiento proyectivo permite continuar el seguimiento a pesar de rotaciones de la cabeza fuera de la frontal a la cámara, la estimación de los parámetros de animación se deteriora cuanto mayor es el giro.

Parte V

Conclusiones

Capítulo 8

Conclusiones y líneas futuras

8.1. Conclusiones

El hilo conductor de la presente tesis es la construcción de un sistema de seguimiento del rostro humano y la cuantificación de sus deformaciones en tiempo real empleando visión por computador. El trabajo se ha dividido en tres partes: seguimiento basado en el color, seguimiento basado en la textura y reanimación de expresiones faciales.

Dado que todo algoritmo de seguimiento tiene condiciones de fallo, empleamos diferentes seguidores funcionando de forma coordinada para obtener un sistema más fiable. Nuestra aproximación al problema es muy parecida al "Enfoque Gradual de la Atención" de Hager y Toyama [92] en cuanto a la arquitectura, pero difiere en cuanto a los algoritmos de seguimiento y análisis facial empleados.

Cuando se utiliza el color de la piel como elemento de seguimiento se obtienen algoritmos rápidos que trabajan fácilmente en tiempo real. El principal problema para estos algoritmos son las variaciones de las condiciones de iluminación de la escena. En la tesis hemos propuesto un procedimiento de normalización del color que mejora el rendimiento de estos seguidores ante condiciones cambiantes de iluminación.

Por otro lado, cuando se emplea la textura del rostro como información para el seguimiento visual, tenemos que hablar de la alineación incremental de imágenes. La solución tradicional es el algoritmo bien conocido de Lucas y Kanade [67]. Esta aproximación es bastante costosa computacionalmente dado que hay que recalcular en cada imagen su matriz Jacobiana respecto a los parámetros de movimiento, $\mathbf{M}$.

En el trabajo de Hager y Belhumeur [40] se propone un método eficiente de seguimiento basado en factorizar la a matriz Jacobiana, $\mathbf{M}$, en $\mathbf{M}_0(\bar{x})\mathbf{\Sigma}(\bar{\mu})$ de forma que los cálculos durante el seguimiento queden limitados, fundamentalmente, al cálculo de la inversa $\mathbf{\Sigma}^{-1}$. Otro algoritmo eficiente, en este caso composicional, el de Baker y Matthews [3], obtiene su eficiencia intercambiando los papeles de la plantilla y la imagen rectificada.

Aunque muy importante en Visión por Computadora, Hager y Belhumeur no resolvieron la factorización del Jacobiano para el caso del modelo de movimiento proyectivo [40]. Recientemente se ha afirmado que una de las limitaciones de esta

técnica de factorización es el que no se pueda emplear con un modelo de movimiento proyectivo [4, 3]. En la presente tesis hemos demostrado cómo se puede realizar la factorización del Jacobiano para el modelo de movimiento proyectivo y como hacerla más eficiente seleccionando los píxeles más informativos para el seguimiento.

También hemos demostrado que la técnica de factorización puede utilizarse para realizar eficientemente el seguimiento de objetos que cambian de apariencia. Nuestra aproximación emplea un modelo de apariencia lineal, como en el *Eigentracking* de Black y Jepson [10] y difiere de los Modelos Activos de Apariencia (AAM) [4] en que no se emplea un modelo de la forma. El algoritmo planteado en la tesis se diferencia del original de Black y Jepson en el uso de un conjunto de plantillas de movimiento precalculadas que hacen más eficiente el proceso de minimización.

Finalmente, una de las posibles aplicaciones del seguimiento basado en la apariencia es la animación dirigida por la actuación de una persona, tanto en 2D como en 3D. En la presente tesis también se ha mostrado cómo se puede emplear la apariencia de las partes más expresivas del rostro en la estimación de los parámetros de animación. El sistema desarrollado, que permite la animación de un modelo 3D del rostro en tiempo real, constituye la prueba principal de nuestra aproximación.

8.2. Aportaciones originales

A continuación se relacionan las principales a aportaciones de la presente tesis:

1. La extensión del algoritmo de constancia del color *Grey World* a secuencias de imágenes. Este nuevo algoritmo de normalización permite el seguimiento basado en el color de la piel con cambios en el color de la iluminación.

2. La introducción en el algoritmo de seguimiento de regiones de Hager y Belhumeur [40] del modelo proyectivo y de un procedimiento de estimación de la rotación y traslación del plano con respecto a la cámara. Suponiendo que la cara está en un plano, el empleo de este seguidor permite rotaciones moderadas de la cabeza fuera del plano de la imagen.

3. Desarrollo de un procedimiento de selección automática de los puntos más informativos de la plantilla de referencia en el algoritmo de Hager y Belhumeur. La reducción en el número de puntos a procesar por el algoritmo permite una reducción del tiempo de cómputo de hasta un 57 %.

4. Planteamiento de un procedimiento eficiente de seguimiento de objetos deformables basado en la factorización del jacobiano.

Finalmente, una de la aportación más importantes, desde el punto de vista práctico, es la implementación de los algoritmos desarrollados en la presente tesis, que permite seguir una cara y reanimar un modelo 3D en tiempo real (ver apéndice D).

8.3. Líneas de investigación abiertas

A pesar de que actualmente disponemos de un sistema que procesa más de 30 imágenes por segundo en el caso del seguidor basado en plantillas y 15 imágenes por segundo para el seguidor basado en la apariencia aún tenemos abiertas varias líneas de trabajo:

- Es posible una reducción importante en el número de operaciones en el algoritmo de seguimiento basado en la apariencia simplemente aprovechando la estructura de las matrices involucradas, dado que muchas de ellas contienen bloques rellenos de ceros.
- Hacer un seguimiento basado en la apariencia eficiente y resistente a los cambios en la iluminación. Hasta ahora simplemente ecualizamos las imágenes.
- Hacer un seguimiento basado en la apariencia eficiente y robusto a oclusiones parciales. Lo que se suele hacer [40, 10, 65] es emplear una métrica robusta en lugar de la norma cuadrática, lo que supone emplear algoritmo basado en IRLS (Iterative Reweighted Least Squares) y tener que recalcular $\mathbf{M}^\top\mathbf{M}$, que es tremendamente ineficiente.
- Encontrar una forma automatizada de relacionar parámetros de seguimiento y animación.

Apéndice A

Desarrollos matemáticos

A.1. Eigentracking de Black y Jepson

En el desarrollo del *Eigentracking* [10] de la sección 4.2.2 se han omitido algunos desarrollos que se han dejado para el presente apéndice. La primera tarea en el *Eigentracking* es encontrar el mínimo de (4.50) con respecto a los parámetros de apariencia $\bar{c}$, y para lograrlo hay que desarrollar $E(\bar{\mu}, \bar{c})$

$$\begin{aligned} E(\bar{\mu},\bar{c}) &= ||\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)\delta\bar{\mu} + \mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t) - \mathbf{B}\bar{c}(t+\delta t)||^2 \\ &= (\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)\delta\bar{\mu} + \mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t) - \mathbf{B}\bar{c}(t+\delta t))^\top \\ &\quad (\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)\delta\bar{\mu} + \mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t) - \mathbf{B}\bar{c}(t+\delta t)) \\ &= 2\delta\bar{\mu}^\top\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top\delta\bar{\mu} + \\ &\quad 2\delta\bar{\mu}^\top\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top\mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t) - \\ &\quad 2\delta\bar{\mu}^\top\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top\mathbf{B}\bar{c}(t+\delta t) + \\ &\quad \mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t)^\top\mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t) - \\ &\quad 2\bar{c}(t+\delta t)^\top\mathbf{B}^\top\mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t) + \\ &\quad 2\bar{c}(t+\delta t)^\top\mathbf{B}^\top\mathbf{B}\bar{c}(t+\delta t) \end{aligned} \tag{A.1}$$

y a continuación derivar con respecto $\bar{c}$

$$\begin{aligned} \frac{\partial E(\bar{\mu},\bar{c})}{\partial\bar{c}} &= -2\mathbf{B}^\top\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)\delta\bar{\mu} \\ &\quad -2\mathbf{B}^\top\mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t) \\ &\quad +2\mathbf{B}^\top\mathbf{B}\bar{c}(t+\delta t). \end{aligned} \tag{A.2}$$

Para encontrar el mínimo igualamos (A.2) a cero y obtenemos el mínimo

$$\bar{c}(t+\delta t) = (\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top[\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)\delta\bar{\mu} + \mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t)], \tag{A.3}$$

y teniendo en cuenta que $\mathbf{B}$ es una matriz ortonormal dado que proviene del PCA de imágenes, tenemos

$$\bar{c} = \mathbf{B}^\top[\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)\delta\bar{\mu} + \mathbf{I}(f(\bar{x},\bar{\mu}_t), t+\delta t)]. \tag{A.4}$$

Para encontrar el mínimo de (4.50) con respecto a $\bar{\mu}$, tendremos que derivar $E(\bar{\mu}, \bar{c})$ con respecto a $\bar{\mu}$

$$\begin{aligned}\frac{\partial E(\bar{\mu}, \bar{c})}{\partial \bar{\mu}} &= 2\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top \mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)\delta\bar{\mu} \\ &\quad +2\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top \mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t) \\ &\quad -2\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top \mathbf{B}\bar{c}(t+\delta t), \end{aligned} \tag{A.5}$$

e igualando a cero A.5

$$\delta\bar{\mu} = -(\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top \mathbf{M}_{et}(\bar{\mu}_t, t+\delta t))^{-1}\mathbf{M}_{et}(\bar{\mu}_t, t+\delta t)^\top [\mathbf{I}(f(\bar{x}, \bar{\mu}_t), t+\delta t) - \mathbf{B}\bar{c}(t+\delta t)]. \tag{A.6}$$

Apéndice B

Entrenamiento automático del seguidor basado en la apariencia

Como ya se vio en el capítulo 6 necesitamos un modelo lineal de la variación de los niveles de gris (debida al movimiento no rígido) para el seguidor basado en la apariencia. El modelo que hemos empleado en los experimentos sobre el *Eigen-tracking factorizado* (ver sección 6.3) es el resultado del Análisis de Componentes Principales (PCA) de imágenes.

En este apéndice detallamos un procedimiento de construcción de modelos de apariencia que no requiere la estimación de tantos parámetros como en el trabajo de De La Torre [65]. Es igualmente automático aunque necesita de más intervención del usuario y al menos de una secuencia de imágenes con movimiento no rígido restringido a una única región del objeto.

B.1. PCA de imágenes

Una imagen I de dimensiones $w \times h$, donde w es el ancho y h es el alto, puede verse como un vector en un espacio N-dimensional, donde $N = w \times h$. Sea $\mathbf{I}$ el vector columna con los niveles de gris de la imagen I recorrida por filas (o por columnas).

El PCA de imágenes trata de construir un modelo lineal de la variación de los niveles de gris en un conjunto de m imágenes ejemplo, $\{I_k\}_{k=1}^{m}$, mediante una base de l vectores, $l << N$, de tal forma que podamos aproximar una imagen cualquiera, $\mathbf{I}$, como

$$\hat{\mathbf{I}} = \bar{\mathbf{I}} + \mathbf{B}\bar{c} = \bar{\mathbf{I}} + \mathbf{B}\mathbf{B}^{\top}(\mathbf{I} - \bar{\mathbf{I}}) \tag{B.1}$$

donde $\bar{\mathbf{I}}$ es la imagen media de los m ejemplos y $\mathbf{B}$ es una matriz ortogonal con los l primeros autovectores de la matriz de covarianzas de los niveles de gris, $\mathbf{C} = \mathbf{A}\mathbf{A}^{\top}$. La matriz $\mathbf{A} = (\mathbf{I}_1 \dots \mathbf{I}_m)$ contiene las m imágenes ejemplo a las que se les ha restado la imagen media previamente.

Dado que $\mathbf{C}$ es una matriz $N \times N$ y típicamente $N = 100 \times 100 = 10000$ píxeles, calcular los autovalores será muy costoso computacionalmente. El truco algebraico que se suele emplear para reducir los cálculos necesarios se debe a Turk y Pentland

[94], y consiste en calcular los autovalores de $\mathbf{A}\mathbf{A}^\top$ a partir de los autovalores de $\mathbf{A}^\top\mathbf{A}$, que es una matriz $m \times m$ (donde m para imágenes suele ser un orden de magnitud inferior a N). Para todo $\bar{x}$, autovector de $\mathbf{A}^\top\mathbf{A}$, con autovalor asociado λ, se cumple:

$$\mathbf{A}^\top\mathbf{A}\bar{x} = \lambda\bar{x}. \tag{B.2}$$

Así que si multiplicando por $\mathbf{A}$ ambos lados de la ecuación (B.2) tendremos

$$\mathbf{A}\mathbf{A}^\top(\mathbf{A}\bar{x}) = \lambda(\mathbf{A}\bar{x}). \tag{B.3}$$

El resultado es que podemos calcular los autovectores de $\mathbf{A}^\top\mathbf{A}$ y multiplicándolos por $\mathbf{A}$ tendremos los autovectores de $\mathbf{A}\mathbf{A}^\top$ que son lo que buscábamos. Los l autovectores de $\mathbf{C}$ con los autovalores asociados mayores serán las columnas de la matriz $\mathbf{B}$.

Una vez tenemos el modelo de PCA de los niveles de gris, constituido por la base del subespacio $\mathbf{B}$ y la imagen media $\bar{\mathbf{I}}$, podemos para cualquier imagen $\mathbf{I}$ podemos obtener su representación en un espacio l-dimensional, $\bar{c}$, como

$$\bar{c} = \mathbf{B}^\top(\mathbf{I} - \bar{\mathbf{I}}). \tag{B.4}$$

B.2. Algoritmo de entrenamiento

Para construir los modelos de PCA necesarios en nuestros experimentos necesitamos un conjunto de imágenes bien alineadas para cada módulo o región a seguir (ojos, boca, nariz, etc). Podríamos recortar a mano cada una de las imágenes de una secuencia de entrenamiento pero dado que tratamos con un número elevado de ejemplos (entre 1000 y 2000 imágenes), y que la selección manual está sujeta a errores, esta solución es inviable. Otra solución, aparecida recientemente en la literatura [65], sería plantear una minimización similar al *Eigentracking* del capítulo 6 en la que no sólo habría que estimar los parámetros de movimiento y apariencia, sino también la base del subespacio lineal. Al tener que estimar al mismo tiempo el modelo, los parámetros de seguimiento y los de apariencia, el resultado es una función de coste con un número elevado de incógnitas. Sin embargo, desde un punto de vista práctico, el procedimiento presentado por De la Torre [65] posee ventajas como la de ser automático y necesitar una única secuencia de imágenes junto con la posición de las regiones a seguir en la primera imagen de la secuencia.

En esta sección presentaremos un algoritmo automático (o al menos con el mismo grado de automatización que en [65]) de construcción de modelos de PCA modulares para el rostro humano. Mediante el empleo del seguidor basado en plantillas y del seguidor de la apariencia, un modelo de movimiento de rotación-traslación-escala y varias secuencias de imágenes con la cara en posición frontal a la cámara, es posible extraer los ejemplos automáticamente.

El algoritmo que hemos empleado en el entrenamiento del *Eigentracking* requiere de más colaboración por parte del usuario que en el algoritmo de De La Torre [65]. Esto es así porque, como veremos a continuación, se necesitan secuencias de imágenes en las que sólo exista movimiento no rígido en determinadas zonas de la cara. El algoritmo para extraer ejemplos automáticamente se expone a continuación:

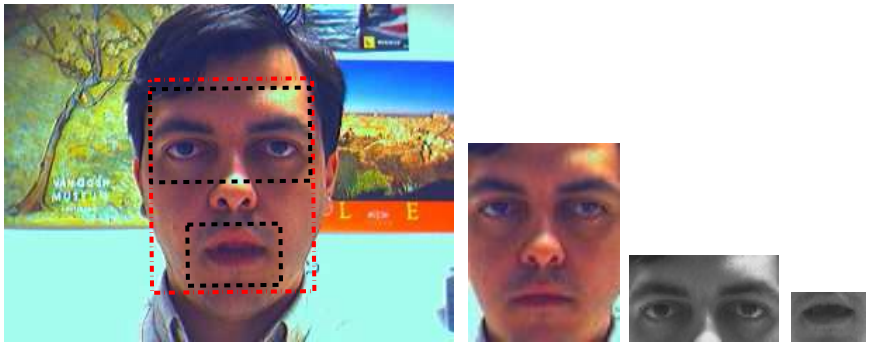


Figura B.1: Regiones de interés sobre la cara. En la imagen más a la izquierda se muestra la posición de las regiones a extraer sobre la cara (línea negra con trazos discontinuos). También se muestra la región de la cara completa en cuyo centro se sitúa el sistema de referencia 2D para todas las regiones (línea roja con trazos y puntos). En la segunda columna se encuentra la imagen estabilizada del rostro completo. En la tercera y cuarta columnas podemos ver las imágenes estabilizadas de los ojos y la boca, respectivamente.

.

- **Entrenamiento:**

  1. Definir un sistema de coordenadas 2D global al rostro en base a una plantilla de referencia para el rostro completo. Todas las regiones de la cara a extraer tendrán coordenadas relativas a esa plantilla global (ver figura B.1).
  2. Para una de las regiones de la cara, a la que llamaremos *Región de Referencia*, adquirir una secuencia de imágenes con movimiento no rígido únicamente en esa región (ver figuras B.2 y B.3).
  3. Para extraer las imágenes de la región de referencia, que presenta movimiento no rígido, emplearemos un seguidor basado en plantillas (y el modelo de movimiento traslación-rotación-escala) de otra región que presenta únicamente movimiento rígido. Como conocemos la posición de una y otra en el sistema de coordenadas 2D global, podremos extraer las imágenes con movimiento no rígido a partir de la posición de la otra región estimada por el seguidor basado en plantillas (ver figuras B.2 y B.3).
  4. Una vez tenemos las imágenes de ejemplo de la región referencia podemos realizar el PCA de imágenes (ver sección B.1), y obtener su modelo de la apariencia.
  5. Elegir una nueva región, R_i y una secuencia en la que aparezca únicamente con movimiento no rígido. Podemos seguir ahora la región de referencia con el seguidor de la apariencia del capítulo 6 y extraer sus imágenes de ejemplo correspondientes.

6. Si todavía existen más regiones sin modelo de PCA ir al paso 4. En otro caso, terminar.

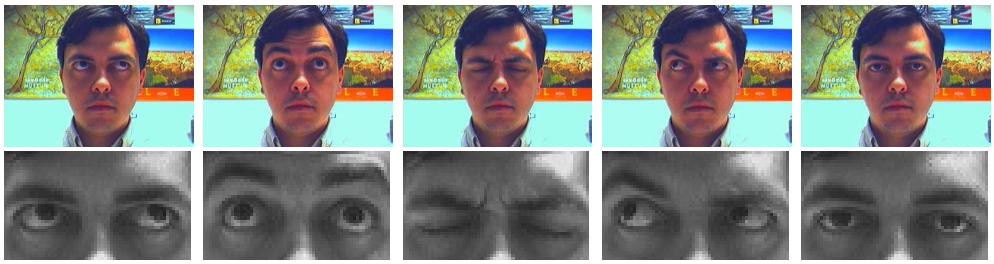


Figura B.2: Secuencia de entrenamiento para los ojos. En la primera fila, algunas imágenes de la secuencia de entrenamiento para el movimiento de los ojos. En la segunda fila, ejemplos de los ojos recortados automáticamente a partir del seguimiento de la boca.

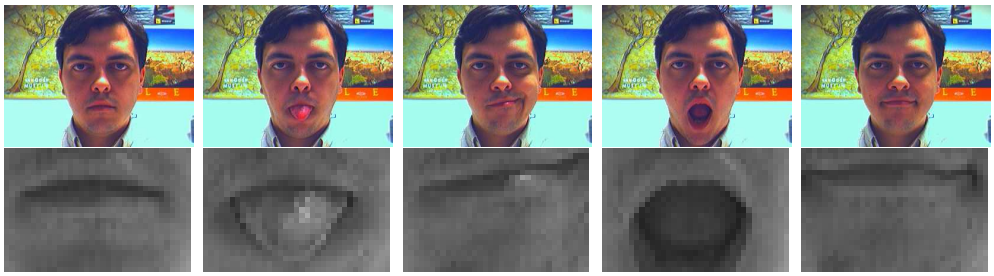


Figura B.3: Secuencia de entrenamiento para la boca. En la primera fila, algunas imágenes de la secuencia de entrenamiento para el movimiento de la boca. En la segunda fila, ejemplos de la boca recortados automáticamente a partir del seguimiento de los ojos.

Si bien es cierto que es necesario marcar en la primera imagen de cada secuencia la posición de la región a seguir, a partir de ahí la extracción de los ejemplos es automática. En la práctica para cada modelo de usuario a construir se captura una única secuencia de imágenes. Al usuario se le pide primero, que sólo realice movimientos en la zona de los ojos y después que sólo lo haga en la zona de la boca. En todo caso también se le pide que mantenga la cabeza lo más frontal y estática posible (aunque el seguidor basado en plantillas resolverá la mayor parte de los movimientos de la misma).

Apéndice C

Modelo gráfico 3D del rostro

Para mostrar los resultados de la reanimación de un modelo 3D del rostro humano partimos del modelo de animación facial basado en músculos de Parke y Waters [101, 74] y cuyo código se encontraba disponible en Internet (en la página web asociada al libro [74]). El modelo de animación permite presentar la cara en cinco modos diferentes (ver figura C.1) con un realismo aceptable en las animaciones y un número no demasiado grande de grados de libertad (21).

Figura C.1: Modos de generación de imágenes del modelo

C.1. Características del modelo original

El modelo original consiste en tres mallas de triángulos: una para cada globo ocular y una para la cara. La superficie de ambos ojos se representa mediante una semiesfera con la posibilidad de aplicarle una textura (ver figuraC.2). Por otro lado la superficie de la cara se genera completa a partir de únicamente uno de los lados de la misma aprovechando la simetría de la misma (ver figuraC.3).

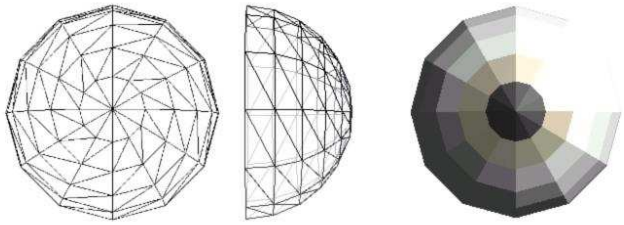


Figura C.2: Superficie del globo ocular

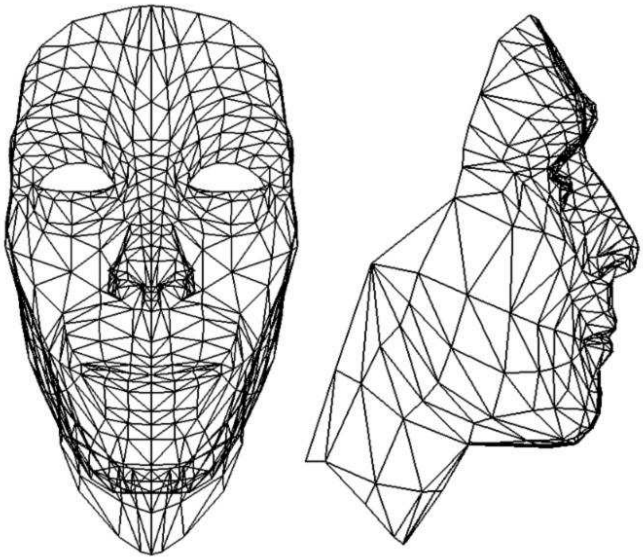


Figura C.3: Superficie facial

El modelo facial permite cuatro tipos de movimientos:

- **Transformaciones globales.** Que consisten en movimientos de afirmación y negación con la cabeza (ninguno más) o rotaciones alrededor de los ejes X e Y.
- **Movimiento de la mandíbula.** Ciertos vértices del modelo giran alrededor del eje de la mandíbula (ver figura C.4). El grado de apertura de la misma se controla mediante un único parámetro.

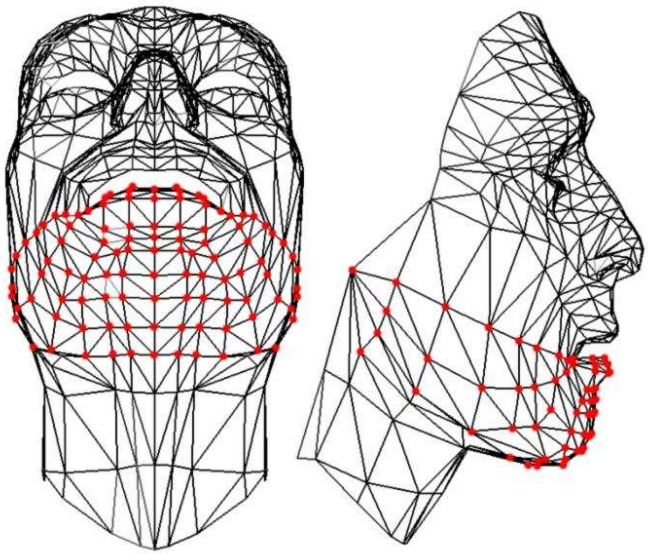


Figura C.4: Movimiento de los vértices pertenecientes a la mandíbula (marcados en rojo)

- **Movimientos musculares.** Permiten la animación facial en sí simulando el comportamiento de los músculos reales de la cara. En el modelo original de Parke y Waters se definen nueve pares de músculos, cada uno con su músculo izquierdo y el correspondiente derecho (ver figura C.5).

C.1.1. Músculos lineales

En el modelo de Parke y Waters un músculo viene definido por dos puntos tridimensionales V_h y V_t, un ángulo α y dos distancias f_s y f_e. Las dos coordenadas tridimensionales definen el vector muscular V, que representa el vector que une la cabeza del músculo V_h (su inserción en el hueso), con el final del músculo V_t (la unión con la piel). El ángulo α y las distancias f_s y f_e determinan la zona de influencia que tiene la transformación. Para simular la contracción de un músculo, se recorren todos los puntos de la superficie facial (vértices de la red de polígonos), calculando para cada punto P su nueva posición P' que viene definida por la ecuación

$$P' = P + a\ k\ r\ \frac{\overline{PV}_h}{|\overline{PV}_h|}. \tag{C.1}$$

El resultado es que cada punto de la superficie se desplaza en dirección a la cabeza del músculo en función de a, k y r. El valor de contracción del músculo viene dado por k. El parámetro a viene definido por

$$a = 1 - \frac{cos(\beta)}{cos(\alpha)} \tag{C.2}$$

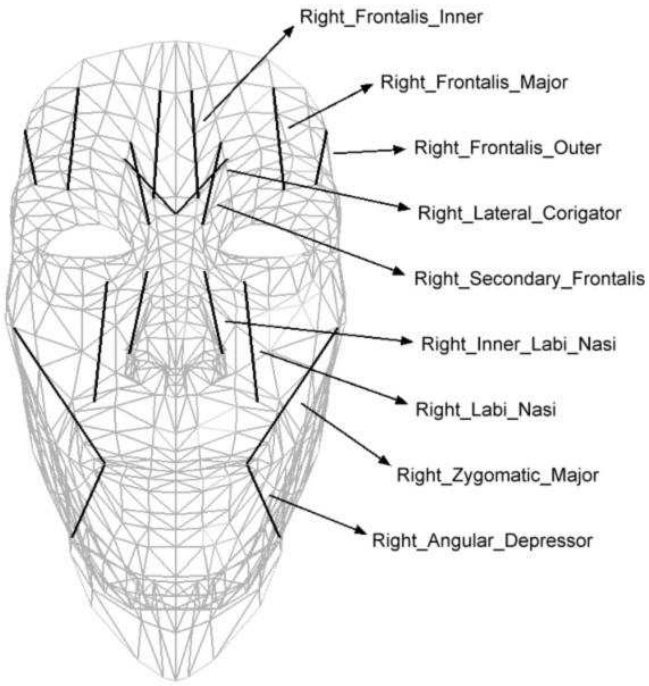


Figura C.5: Músculos sobre el rostro. Únicamente se escribe el nombre de los del lado derecho

donde β es el ángulo formado entre el vector muscular V y el vector que une la cabeza del músculo y P. El parámetro a se hace mayor cuanto menor es el ángulo β y haciéndose máximo cuando $\beta = 0$, o lo que es lo mismo, cuando el punto P se encuentra sobre la línea que une V_h y V_t.

Por último el parámetro r viene definido por

$$r = i \begin{cases} 0 & si\ |\overline{PV}_h| \notin (0, f_e] \vee cos(\alpha) > cos(\beta) \\ cos(\frac{\frac{\pi}{2}(|\overline{PV}_h| - f_s)}{f_e - f_s}) & si\ |\overline{PV}_h| \in [f_s, f_e] \wedge cos(\alpha) \leq cos(\beta) \\ 1 & si\ |\overline{PV}_h| \notin (0, f_s) \wedge cos(\alpha) > cos(\beta) \end{cases} \quad \text{(C.3)}$$

y divide el espacio en tres regiones. Las primera región, donde el valor de r es máximo ($r = 1$), se define mediante la intersección de la esfera con centro V_h y radio f_s con el cono formado por la revolución alrededor de V de una línea que mantiene un ángulo α con V y cuyo vértice es V_h. La segunda región, donde r toma un valor en función de la proximidad de P a la primera región, es la prolongación del cono que forma la primera región hasta la esfera con centro V_h y radio f_e. La última región es la exterior a las dos primeras y r toma el valor cero, lo que significa que el músculo no afectará a P.

Cuando aplicamos una contracción de valor k a un músculo lineal se desplazarán en mayor o menor medida todos los puntos de la malla de la cara en dirección a la cabeza del mismo. La contracción afectará a todos los puntos de la malla que se encuentren dentro de la primera y segunda regiones (un "cono" de base esférica)

dependiendo de su distancia a la cabeza del músculo. Y por último el movimiento de los puntos de la malla también dependerá de su distancia al eje del músculo (menor movimiento cuanto más alejados).

C.2. Modificaciones introducidas

Dado que al modelo de partida le faltaban algunas características que consideramos fundamentales lo modificamos en consecuencia:

- **Transformaciones globales**. Se ha introducido la posibilidad de realizar rotaciones en los tres ejes (ver figura C.6) y translaciones en los mismos (ver figura C.7).

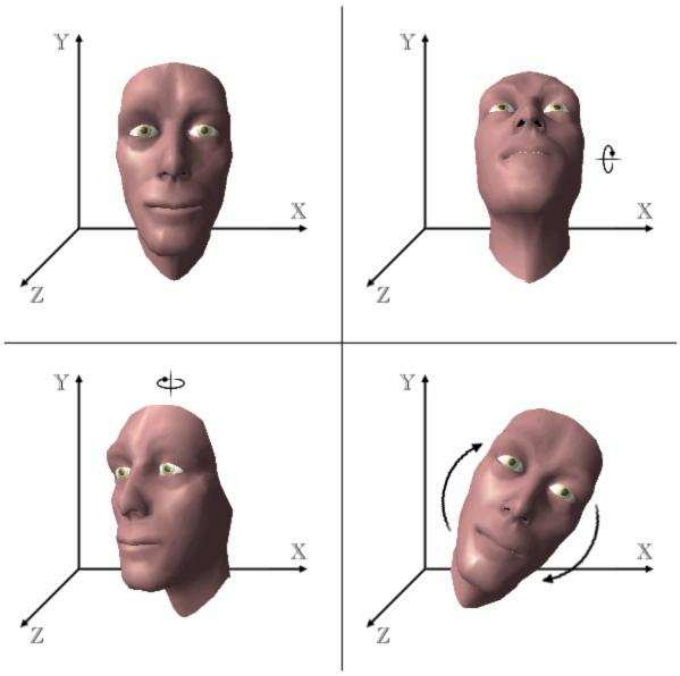


Figura C.6: Rotaciones en los tres ejes de la cabeza.

- **Giro independiente de ambos ojos**. En el modelo original no se contemplaba la posibilidad del giro de los ojos en sus órbitas. Se ha añadido las rotaciones naturales (alrededor del eje horizontal y del vertical) en ambos ojos (ver figura C.8).

- **Movimiento de los párpados**. El parpadeo es una acción fundamental en cualquier animación del rostro humano. El modelo de partida no lo permitía y hubo que desarrollar un procedimiento similar al de la mandíbula para permitir su movimiento (ver figura C.9).

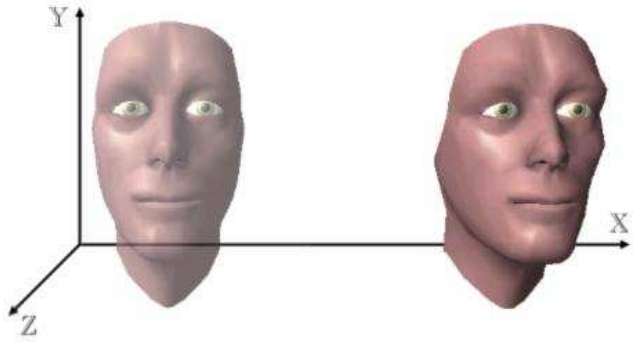


Figura C.7: Ejemplo de traslación posible con el nuevo modelo.

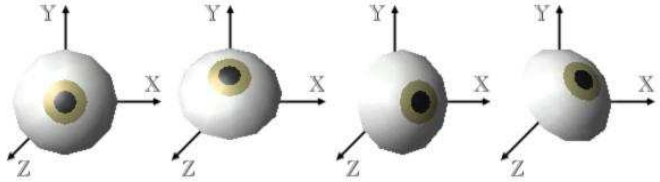


Figura C.8: Rotaciones de los ojos

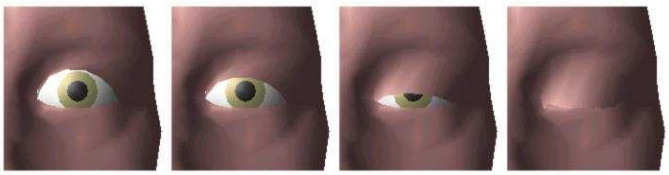


Figura C.9: Ejemplo de movimiento de los párpados

- **Músculos orbiculares de los labios**. La contracción de este músculo desplaza los labios hacia el centro de la boca. Cuando la boca está abierta la forma de los labios describe una forma más redondeada, y en caso de estar cerrada los labios se comprimirán los unos con los otros acercando sus comisuras (ver figura C.10). Este tipo de músculo es imprescindible si se quiere simular un beso, por ejemplo.

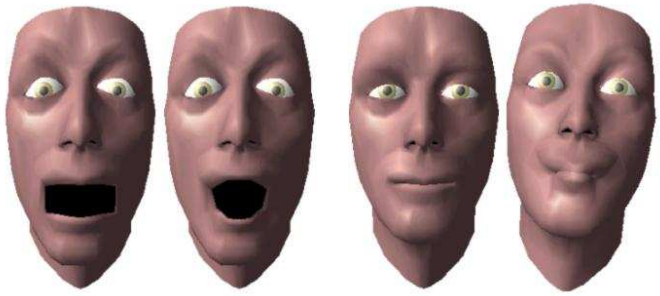


Figura C.10: Efecto de los músculos orbiculares de los labios

- **Interpolación de expresiones clave**. Cuando hay que generar animaciones del rostro para realizar experimentos de análisis de expresiones faciales se necesita generar unos cuantos cientos de imágenes en una secuencia. La definición de las expresiones del rostro imagen a imagen es tedioso y sujeto a errores (discontinuidades en el movimiento de las diferentes partes del rostro).

  Mediante las expresiones clave (ver figura C.11), se definen expresiones (un conjunto de parámetros: contracción de músculos, posición, etc) por las cuales la animación debe pasar, generándose de forma automática las expresiones intermedias. Dadas dos expresiones clave y una serie de parámetros, es posible realizar una interpolación para obtener todas las expresiones intermedias. Hemos añadido esta posibilidad al modelo de Parke y Waters para generar secuencias de imágenes con animaciones faciales de cientos de imágenes simplemente definiendo una veintena de expresiones faciales.

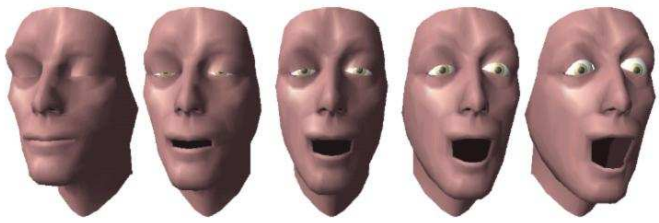


Figura C.11: Interpolación entre dos expresiones clave

El resultado de las modificaciones introducidas es un modelo de animación del rostro humano con 32 grados de libertad (ver cuadro C.1). Estos parámetros reflejarán la traslación (3), la rotación (3), el estado de cada músculo lineal (18), el

grado de abertura de la boca (1), la contracción de los labios (1), la posición de los párpados (2) y la orientación de cada ojo (4).

	Nombre	Rango recomendado
1	traslación X	[-60, +60] Desplazamiento sobre el eje X
2	traslación Y	[-50, +50] Desplazamiento sobre el eje Y.
3	traslación Z	[-200, -12] Desplazamiento sobre el eje Z.
4	rotación X	(0, 360) Ángulo de giro en grados sobre el eje X.
5	rotación Y	(0, 360) Ángulo de giro en grados sobre el eje Y.
6	rotación Z	(0, 360) Ángulo de giro en grados sobre el eje Z.
7	contracción músculo 1	[0, 6] Según crece contrae más el músculo
...	...	...
25	abertura boca	[0, 15] Según crece abre en mayor ángulo la boca.
26	contracción labios	[0, 6] Según crece se contrae más el esfínter.
27	párpado derecho	[-0.4, 1] Según crece se cierra mas el párpado derecho.
28	párpado Izquierdo	[-0.4, 1] Según crece se cierra mas el párpado Izquierdo.
29	rotación X ojo derecho	(-45, 45) Ángulo de giro sobre el eje X del ojo derecho.
30	rotación Y ojo derecho	(-45, 45) Ángulo de giro sobre el eje Y del ojo derecho.
31	rotación X ojo izquierdo	(-45, 45) Ángulo de giro sobre el eje X del ojo izquierdo.
32	rotación Y ojo izquierdo	(-45, 45) Ángulo de giro sobre el eje Y del ojo izquierdo.

Cuadro C.1: Grados de libertad del modelo y los rangos de valores asociados.

Apéndice D

Software desarrollado

En la investigación en Visión por Computador uno de los aspectos más importante es el software. Se proponen modelos que hay que probar mediante experimentos. Las pruebas se realizan sobre las implementaciones de los algoritmos y son las que permiten discernir si un algoritmo funciona mejor que otro. En el marco de la presente tesis doctoral se han escrito una serie de bibliotecas de programación para ayudar en las pruebas de los algoritmos de análisis facial.

El software se ha escrito en C++ intentando siempre conseguir la máxima transportabilidad (entre MS Windows y GNU/Linux por ejemplo). De esta forma se utiliza la biblioteca estándar de plantillas (STL) del estándar de C++, bibliotecas de programación de terceros que permitan la transportabilidad (a nivel de código fuente) y se ha diseñado de forma que los algoritmos de visión no queden expuestos directamente al código dependiente de la plataforma (como los sistemas de ventanas y la captura de vídeo). En la figura D.1 se esquematizan las dependencias entre las diferentes bibliotecas que conforman el software, así como las dependencias con otras bibliotecas de terceros (de libre distribución).

El software se encuentra orientado al procesamiento de secuencias de imágenes en tiempo real, por lo que cubre los diferentes aspectos implicados en el proceso:

- Definición de la interfaz para el acceso a los datos de una imagen (paintlib y libimprocess).
- Algoritmos de procesamiento de imagen (libimprocess).
- Definición de una interfaz C++ de diferentes algoritmos de cálculo numérico y álgebra lineal (libnumeric).
- Definición de una interfaz común para el acceso a los dispositivos de captura de imágenes (libvideosrc).
- Definición de una interfaz software independiente de la plataforma que permita la presentación de resultados de los algoritmos de visión (libsyswindow).
- Definición e implementación de los algoritmos de seguimiento visual (trackinglib).

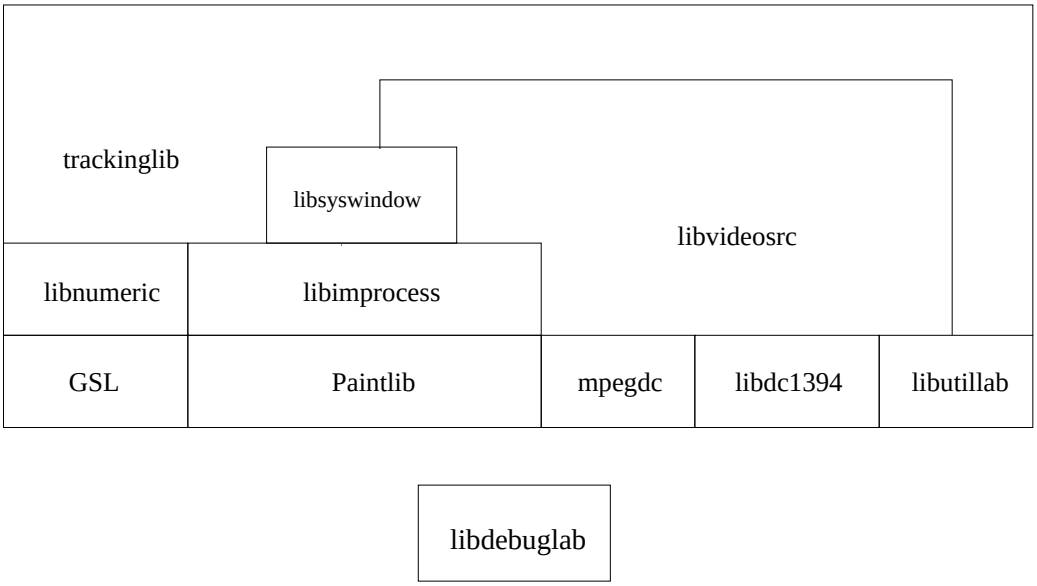


Figura D.1: Diagrama de capas del software. Cuando una capa se encuentra sobre otra significa que la primera utiliza a la última. La biblioteca libdebuglab, se utiliza en todas las demás para ayudar a la depuración del código.

D.1. Procesamiento de imagen: libimprocess.

La biblioteca de procesamiento de imagen emplea la interfaz de imagen definida en la clase CBmp de paintlib [50]. Paintlib es una biblioteca C++ transportable (usada sin problemas en GNU/Linux y Windows), que permite leer de disco multitud de formatos de ficheros gráficos.

En libimprocess se define *VideoBmp*, que será la clase que se emplea en las demás bibliotecas para manipular los datos de una imagen. En todo el sistema siempre se trata con imágenes en color de 32 bits por pixel (R,G,B y transparencia, aunque esta última no se utiliza).

En la biblioteca se han programado diferentes algoritmos de procesado de imagen: detección de bordes, generación de una pirámide gaussiana, "warping" afín y perspectivo, ecualización del histograma, etc.

Para acelerar todo lo posible el procesamiento de imagen se han encapsulado algunos de los algoritmos proporcionados por la biblioteca de procesamiento de imagen de Intel (IPL) [54] en C++. Esta biblioteca es extremadamente rápida ya que emplea SIMD (Single Instruction Multiple Data) con las instrucciones MMX y SSE. Además se encuentra disponible tanto para GNU/Linux como para MS Windows.

D.2. Algoritmos numéricos y álgebra lineal: libnumeric.

La biblioteca de matemáticas encapsula en C++ algunas de las funciones de GSL [44]. GSL tiene implementados multitud de algoritmos de análisis numérico, álgebra lineal (soporta BLAS), estadística, física, optimización y un largo etcétera. La ventaja fundamental es que se ha escrito en C y el propósito es que sea altamente transportable entre arquitecturas diferentes.

Una biblioteca en C++ que se interpone entre los algoritmos de seguimiento visual y GSL es necesaria para aislar a los primeros de un hipotético cambio de biblioteca de análisis numérico.

D.3. Captura de imagen: libvideosrc.

Para la captura de imagen se define una clase abstracta *VideoSource* que establece la interfaz para todas las fuentes de vídeo a utilizar. Las fuentes de vídeo concretas se implementan en clases derivadas de *VideoSource*.

La biblioteca actualmente tiene disponibles las siguientes fuentes de vídeo:

- Independientes de la plataforma:
  - *VideoSourceMpeg* para procesar vídeos en formato mpeg. Para decodificar el flujo de datos mpeg se emplea la biblioteca C++ mpegdc [102]. Como la decodificación es un poco lenta, está previsto utilizar la SMPEG [47] que permite mayores velocidades de descompresión y es igualmente transportable.
  - *VideoSourceImageFiles* para procesar secuencias de vídeo almacenadas en disco como imágenes individuales (BMP, JPG, PNG, PGM, TIFF, PCX, etc.), empleando para leerlas los decodificadores presentes en Paintlib.
- Para GNU/Linux:
  - *Video4LinuxSource* para capturar de dispositivos cuyo driver siga la interfaz video4linux. Funciona a 25 imágenes por segundo empleando tarjetas capturadoras de televisión con driver BTTV.
  - *VideoSourceDC1394* para capturar de cámaras digitales IEEE 1394 (sin compresión). Por el momento los driver IEEE1394 en Linux no se consideran estables, aunque la gente los utiliza sin excesivos problemas para capturar imágenes a 30 cuadros por segundo a resolución 640x480 YUV422. Esta clase no utiliza directamente los drivers a través de sus interfaces "ioctl" sino que se utiliza la biblioteca libdc1394 [43] para el control y captura desde cámaras firewire.
- Para MS Windows:

- *VideoSourceWinDC1394* para capturar de cámaras digitales IEEE1394 (Sin compresión). MS Windows 98, ME, 2000 y NT, en principio tienen implementado el soporte para IEEE1394 (para tarjetas compatibles OHCI) pero no existe un driver de serie para poder controlar y capturar desde cámaras digitales. En esta clase se utiliza un driver para esta cámaras desarrollado en el CMU [45].

Todas las fuentes de vídeo leen sus parámetros de un fichero de configuración común y se crean a través de la clase *VideoSourceFactory*. Por otro lado todas las fuentes de vídeo pueden tener almacenada en el fichero de configuración su matriz de intrínsecos asociada.

D.4. Presentación de resultados: libsyswindow.

Con libsyswindow se pretende proporcionar a los algoritmos de visión, una abstracción del sistema de ventanas en la que poder presentar imágenes y primitivas gráficas, a través de una interfaz común independiente del sistema de ventanas en el que trabajemos.

Para ello se cuenta con varias clases:

- *SysWindow* clase abstracta que define la interfaz de toda ventana.
- *GraphicWindow* clase abstracta que define la interfaz de una ventana que permite visualizar imágenes y primitivas gráficas.
- *WindowsSystemBase* clase abstracta que define las operaciones básicas a realizar sobre un sistema de ventanas: iniciar, procesar eventos y finalizar.

En la biblioteca se encuentran implementaciones concretas de las diferentes clases tanto para GTK+ como para Win32.

Es posible que en un futuro cercano se modifique sustancialmente esta biblioteca para emplear SDL (Simple DirectMedia Layer) [46]. Esta biblioteca es altamente transportable (GNU/Linux, Windows, MacOS, MacOS X, BeOS, etc) y proporciona servicios de bajo nivel para la realización de juegos: acceso al CD-ROM, al sonido, compatibilidad con OpenGL, acceso al sistema de ventanas y eventos, y por último hilos de ejecución. Todo de una forma transparente y desarrollado para obtener el máximo rendimiento debido a que se emplea principalmente para portar juegos de una plataforma a otra.

D.5. Seguimiento visual: trackinglib.

En esta biblioteca se encuentran los algoritmos de seguimiento visual. La clase principal es *BasicTracker*, que es abstracta y establece la interfaz de todos los

algoritmos de seguimiento. El estado del objeto seguido se mantiene en la clase *TargetState*. Y las estadísticas de la ejecución de los algoritmos se recogen en objetos de la clase *StatisticsData* y se almacenan en disco mediante un objeto *StatisticsWriter*.

Bibliografía

[1] Jörgen Ahlberg. Using the active appearence algorithm for face and facial feature tracking. In *Proc. of 2nd Int. Workshop on Recognition, analisys and tracking of faces and gestures in real time systems, RATFG-RTS'01*, pages 68–72, 2001.

[2] S. Baker, R. Gross, and I. Matthews. Lucas-kanade 20 years on: A unifying framework: Part 3. Technical Report CMU-RI-TR-03-35, Robotics Institute, Carnegie Mellon University, November 2003.

[3] S. Baker and I. Matthews. Lukas-kanade 20 years on: A unifiying framework. *International Journal of Computer Vision (IJCV)*, 56(3):221–255, 2004.

[4] Simon Baker and Ian Matthews. Equivalence and efficiency of image alignment algorithms. In *Proc. of Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–1090–I–1097. IEEE, 2001.

[5] B. Bascle and A. Blake. Separability of pose and expression in facial tracing and animation. In *Proc. of International Conference on Computer Vision*, pages 323–328. IEEE, 1998.

[6] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):711–720, July 1997.

[7] Luis M. Bergasa. *Seguimiento facual mediante visión artificial, orientado a la ayuda a la movilidad*. PhD thesis, Universidad de Alcalá de Henares, Madrid, Spain, 1999.

[8] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proc. of Conference on Computer Vision and Pattern Recognition*, pages 232–237. IEEE, 1998.

[9] M. J. Black and Y. Yacoob. Recognizing facial expressions in image sequences using local parameterized models of image motion. *Int. Journal of Computer Vision*, 25(1):23–48, 1997.

[10] Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision (IJCV)*, 26(1):63–84, 1998.

[11] Jean Yves Bouguet. Camera calibration toolbox for matlab.

[12] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. In *Proc. of Workshop on applications of Computer Vision, WACV'98*, pages 214–219. IEEE, 1998.

[13] Matthew Brand. Voice puppetry. In *Proc. of SIGGRAPH*, pages 21–28. ACM, 1999.

[14] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video rewrite. In *Proc. of SIGGRAPH'1997*, pages 353–360. ACM, 1997.

[15] G. Buchsbaum. A spatial processor model for object colour perception. *Journal of the Fanklin Institute*, 310:1–26, 1980.

[16] Ian Buck, Adam Finkelstein, Charles Jacobs, Allison Klein, David H. Salesin, Joshua Seims, Richard Szeliski, and Kentaro Toyama. Performance-driven hand-drawn animation. In *Proc. of Int. Symposium on Non Photorealistic Animation and Rendering, NPAR'2000*, pages 101–108, 2000.

[17] Y. Cheng. Mean shift, mode seeking and clustering. In *Trans. Pattern Analysis and Machine Intelligence (PAMI)*, volume 17, pages 790–799. IEEE, 1995.

[18] J. Cohn, T. Kanade, T. Moriyama, Z. Ambadar, J. Xiao, J. Ga, and H. Imamura. A comparative study of alternative facs coding algorithms. Technical report, Robotics Institute, Carnegie Mellon University, November 2001.

[19] D. Comaniciu, V. Ramesh, and P. Meer. Real-tiem tracking of non-rigid objects using mean shift. In *Proc. of Conference on Computer Vision and Pattern Recognition*, pages 142–149. IEEE, 2000.

[20] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):11–20, 1994.

[21] T. Cootes, G.J. Edwards, and C. Taylor. Active appearance models. In *Proc. European Conference on Computer Vision*. Springer-Verlag, 1998.

[22] T.F. Cootes and P. Kittipanya-ngam. Comparing variations on the active appearance model algorithm. In *Proc. British Machine Vision Conference*, volume 2, pages 837–846, 2002.

[23] T.F. Cootes and C.J. Taylor. Statistical models of appearance for computer vision. Technical report, Imaging Science and Biomedical Engineering, University of Manchester, 2001.

[24] Trevor J. Darrell, Irfan Essa, and Alex P. Pentland. Task-specific gesture analysis in real-time using interpolated views. *Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 18(12):1236–1242, December 1996.

[25] F. De la Torre and M. J. Black. Dynamic coupled component analysis. In *Proc. of Conference on Computer Vision and Pattern Recognition*, volume II, pages 643–650. IEEE, 2001.

[26] F. de la Torre and M.J. Black. Robust parameterized component analysis. In *Proc. European Conference on Computer Vision (4)*, volume 2353 of *Lecture Notes on Computer Science*, pages 653–669. Springer, 2002.

[27] F. Dellaert and R. Collins. Fast image-based tracking by selective pixel integration. In *ICCV99 Workshop on frame-rate applications*. IEEE, 1999.

[28] M. D'Zmura and P. Lennie. Machanisms of colour constancy. *Journal of the Optical Society of America A*, 3:1662–1672, 1986.

[29] Tony Ezzat, Gadi Geiger, and Tomaso Poggio. Trainable videorealistic speech animation. In *Proc. of SIGGRAPH*. ACM, 2002.

[30] C. Thorpe F. Dellaert and S. Thrun. Super-resolved texture tracking of planar surface patches. In *Proceedings Intelligent Robots and Systems*, pages 197–203. IEEE, 1998.

[31] Graham Finlayson and Gerald Schaefer. Single surface colour constancy. In *7th Color Image Conference*, Scottsdale, Arizona, USA, November 1999.

[32] Graham D. Finlayson. Computational colour constancy. In *Proc. of International Conference on Pattern Recognition*, pages 191–196. IEEE, 2000.

[33] B.J. Frey, A. Colmenarez, and T.S. Huang. Mixtures of local linear subspaces for face recognition. In *Proc. of Conference on Computer Vision and Pattern Recognition*, pages 32–37, June 1998.

[34] S. A. Shafer G. J. Klinker and T. Kanade. A physical approach to color image understanding. *International Journal of Computer Vision (IJCV)*, 4(1):7–38, 1990.

[35] Andrew Gee and Roberto Cipolla. Fast visual tracking by temporal consensus. *Image and Vision Computing*, 14(2):105–114, 1996.

[36] Z. Ghahramani and G.E. Hinton. The em algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto, 1997.

[37] Taro Goto, Sumedha Kshirsagar, and Nadia Magnenat-Thalmann. Automatic face cloning and animation. *Signal Processing Magazine*, 18(3):17–25, May 2001.

[38] Bernt Shiele Graham D. Finlayson and James L. Crowley. Comprehensive colour image normalization. In *Proc. European Conference on Computer Vision (1)*, volume 1406 of *Lecture Notes on Computer Science*, pages 475–490. Springer, 1998.

[39] MPEG-4 Video Group. Coding of audio-visual objects: video. *ISO/IEC*, JTC1/SC29/WG11 N2202, 1998.

[40] Gregory Hager and Peter Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 20(10):1025–1039, 1998.

[41] Gregory D. Hager and PeterÑ. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. In *Proc. of International Conference on Computer Vision and Pattern Recognition, CVPR'96*, pages 403–410. IEEE, 1996.

[42] XinWen Hou, Stan Z. Li, HogJiang Zhang, and QianSheng Cheng. Direct appearance models. In *Proc. of Conference on Computer Vision and Pattern Recognition*. IEEE, 2001.

[43] http://sourceforge.net/projects/libdc1394/. libdc1394: biblioteca escrita en c para el control y captura de cámaras digitales (sin compresión) sobre ieee 1394 en linux.

[44] http://sources.redhat.com/gsl. Gsl: Gnu scientific library.

[45] http://www.cs.cmu.edu/ iwan/1394/. wind1394: Driver para el control y captura de una cámara ditgital ieee1394 sobre ms windows (98, me, nt y 2000).

[46] http://www.libsdl.org. Sdl: Simple directmedia layer.

[47] http://www.lokigames.com/development/. Smpeg: Sdl mpeg player library.

[48] http://www.measurand.com/. Measurand inc, mechanical motion capture systems.

[49] http://www.motionanalysis.com/. Motion analysis corporation, electro-optical motion capture systems.

[50] http://www.paintlib.de. Paintlib: biblioteca escrita en c++ para el manejo de diferentes formatos de almacenamiento de imágenes.

[51] http://www.polhemus.com/. Polhemus, magnetic motion capture systems.

[52] http://www.ptiphoenix.com. Phoenix technologies inc., active-optical motion capture systems.

[53] http://www.vicon.com/. Vicon, optical motion capture systems.

[54] http://developer.intel.com/software/products/perflib/ipl/index.htm Intel Corp. Ipl: Intel image processing library.

[55] M. Isard and A. Blake. Condensation–conditional density propagation for visual tracking. *International Journal of Computer Vision (IJCV)*, 29(1):2–28, 1998.

[56] Michael Isard. *Visual Motion Analisys by Probabilistic Propagation of Conditional Density*. PhD thesis, University of Oxford, 1998.

[57] J. Schwerdt J. L. Crowley. Robust tracking and compression for video communication. In *Proc of the Int. Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time (RATFG'99)*, pages 2–9. RATFG'99, 1999.

[58] Allan Jepson, David Fleet, and Thomas El-Maraghi. Robust online appearance models for visual tracking. In *Proc. of Conference on Computer Vision and Pattern Recognition*, volume I, pages 415–422. IEEE, 2001.

[59] Weier Lu Jie Yang and Alex Waibel. Skin-color modeling and adaptation. In *Proc. of Third Asian Conference on Computer Vision, ACCV'98*, volume 2, pages 687–694. Springer-Verlag, 1997.

[60] Weier Lu Jie Yang and Alex Waibel. Skin-color modeling and adaptation. In *Proceedings Third Asian Conference on Computer Vision Vol. II*, pages 142–147, 1998.

[61] Jun-yong and Ulrich Neumann. Expression cloning. In *Proc. of SIGGRAPH*, pages 277–288. ACM, 2001.

[62] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 24(7):996–100, 2002.

[63] A. Kannan, N. Jojic, and B.J. Frey. Fast transformation-invariant factor analysis. In *Advances in Neural Information Processing Systems 2002*, volume 15. MIT-Press, 2003.

[64] Marco La Cascia, Stan Sclaroff, and Vassilis V. Athitsos. Fast, reliable head tracking under varying illumination: An approach based on robust registration of texture-mapped 3d models. *Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 22(4):322–336, April 2000.

[65] Fernando De la Torre and Michael J. Black. Robust parameterized component analysis: theory and applications to 2d facial appearance models. *Computer Vision and Image Understanding*, 91(1–2):53–71, July-August 2003.

[66] H. Lee. Method for computing the scene illuminant chromaticity from specular highlights. *Journal of the Optical Society of America A*, 3:1694–1699, 1986.

[67] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of Imaging Understanding Workshop*, pages 121–130, 1981.

[68] Birgitta Martinkauppi Maricor Soriano and Sami Huovinen. Skin detection in video under changing illumination conditions. In *Proc. of International Conference on Pattern Recognition*, pages 839–842. IEEE, 2000.

[69] Birgitta Martinkaupi. *Face colour under varying illumination analysis and applications*. PhD thesis, Department of Electrical and Information Engineering and Infortech Oulu, University of Oulu, Finland, 2002.

[70] Hans J. Andersen Miritz Störring and Erik Granum. Skin colour detection under changing lighting conditions. In *7th Symposium on Intelligent Robotics Systems*, pages 187–195, 1999.

[71] Hans J. Andersen Miritz Störring and Erik Granum. Estimation of the illuminant colour from human skin colour. In *Proc. of International Conference on Automatic Face and Gesture Recognition*, pages 64–69. IEEE, 2000.

[72] Hans J. Andersen Moritz Störring and Erik Granum. Physics-based modelling of human skin colour under mixed illuminants. *Robotics and Autonomous Systems*, 35:131–142, 2001.

[73] S.Ñayar, H. Murase, and S.A. Nene. Parametric appearance representation. In S.K. Nayar and T. Poggio, editors, *Early visual learning*, pages 131–160. Oxford University Press, 1996.

[74] Frederick I. Parke and Keith Waters. *Computer Facial Animation*. AK Peters Ltd, 1996.

[75] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge (UK) and New York, 2nd edition, 1992.

[76] A.D. Jepson R. Gershon and J.K. Tsotsos. From [r,g,b] to surface reflectance: Computing color constant descriptors in images. In *Proc. of International Joint Conference on Artificial Intelligence*, pages 755–758, 1987.

[77] C. Rasmussen and G. Hager. An adaptative model for tracking objects by color alone. Technical Report DCS-TR-1200, Yale University, 1998.

[78] S. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2001.

[79] K. Sbottka and I. Pitas. Segmentation and tracking of faces in color images. In *Proc. of the Second International Conference on Automatic Face and Gesture Recognition*, pages 236–242, 1996.

[80] A. Shashua, A. Levin, and S. Avidan. Manifold pursuit: a new approach to appearance based recognition. In *Proc. of International Conference on Pattern Recognition*, volume III, pages 590–594, Quebec, Canada, August 2002. IEEE.

[81] Heung-Yeung Shum and Richard Szeliski. Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision (IJCV)*, 36(2):101–130, 2000.

[82] G. Simon, A. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *Proc. International Symposium on Augmented Reality*, October 2000.

[83] Wladyslaw Skarbek and Andreas Koschan. Colour image segmentation - a survey. Technical report, Technical University of Berlin, Germany, 1994.

[84] D. Skokaj, H. Bischof, and A. Leonardis. A robust pca algorithm for building representations from panoramic images. In *Proc. European Conference on Computer Vision (4)*, volume 2353 of *Lecture Notes on Computer Science*, pages 761–775. Springer, 2002.

[85] Jean-Luc Dugelay Stéphane Valente, Ana C. Andrés Del Valle. Analysis and reproduction of facial expressions for realistic communicating clones. *Journal of VLSI Signal Processing*, 29:41–49, 2001.

[86] Peter F. Sturm and Stephen J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proc. of Conference on Computer Vision and Pattern Recognition*, volume I, pages 432–437. IEEE, 1999.

[87] Irfan Essao Sumit Basu and Alex Petland. Motion regularization for model-based head tracking. Technical Report 362, M.I.T. Media Laboratory Perceptual Computing Section, 1995.

[88] Demetri Terzopoulos and Keith Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 15(6), June 1997.

[89] Y. Tian, T. Kanade, and J. Cohn. Recognizing action units for facial expression analysis. *Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 23(2):97–115, Frebruary 2001.

[90] Michael E. Tipping and Christopher M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[91] L. Torresani, D. Yang, G. Alexander, and C. Bregler. Tracking and modelling non-rigid objects with rank constraints. In *Proc. of Conference on Computer Vision and Pattern Recognition*. IEEE, 2002.

[92] K. Toyama and G.D. Hager. Incremental focus of attention for robust visual tracking. In *Proc. of Conference on Computer Vision and Pattern Recognition*, pages 189–195. IEEE, 1996.

[93] Kentaro Toyama. Prolegomena for robust face tracking. Technical Report MSR-TR-98-65, Microsoft Research, November 1998.

[94] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neurosience*, 3(1), 1991.

[95] F. Lerasle V. Ayala, J.B. Hayet and M. Devy. Visual localization of a mobile robot in indoor environments using planar landmarks. In *Proceedings Intelligent Robots and Systems*, pages 275–280. IEEE, 2000.

[96] S. Valente and J.-L. Dugelay. Analysis and reproduction of facial expressions for communicating clones. In *Proc. of Workshop on Multimedia Signal Processing*. IEEE, 1999.

[97] S. Valente and J. L. Dugelay. Face tracking and realistic animations for telecommunicant clones. *Multimedia Magazine*, pages 34–42, Frebuary 2000.

[98] Stephan Valente and Jean-Luc Dugelay. A visual analysis/synthesis feedback loop for accurate face tracking. *Signal Processing: Image Communications*, 16:585–608, 2001.

[99] Vezhnevets Vladimir and Dead Moroz. Camera calibration toolbox for matlab enhancement.

[100] A. Waibel, M. T. Vo, and P. Duchnowski. Multimodal interfaces. *Artificial Intelligence Review*, 10:299–319, 1996.

[101] Keith Waters. A muscle model for animating three-dimensional facialexpression. In *Proc. of SIGGRAPH*, volume 21, pages 17–24. ACM, 1987.

[102] http://www.cs.wayne.edu/ dil/research/mdc/ Wayne State University. Mpeg developing classes.

[103] Y. Wu, Q. Liu, and T.S. Huang. Robust realtime hand localization by self-organizing color segmentation. In *Proc of the Int. Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time (RATFG'99)*, pages 161–166. RATFG'99, 1999.

[104] Stephen J. Mackenna Yogesh Raja and Shaogang Gong. Colour model selection and adaptation in dynamic scenes. In *Proc. European Conference on Computer Vision (1)*, volume 1406 of *Lecture Notes on Computer Science*, pages 460–474. Springer, 1998.

[105] Benjamin D. Zarit. Skin detection in video images. Technical Report VisLab-99-01, Wrigth State University, Ohio, USA, February 1999.